ESD-TR-77-137

MTR-3349

GEOGRAPHIC DATA DISPLAY IMPLEMENTATION

JUNE 1977

Prepared for

DEPUTY FOR DEVELOPMENT PLAN
ELECTRONIC SYSTEMS DIVISION
AIR FORCE SYSTEMS COMMAND
UNITED STATES AIR FORCE
Hanscom Air Force Base, Bedford, Massachusetts

Approved for public release;
distribution unlimited.

Project No. 7090
Prepared by
THE MITRE CORPORATION
Bedford, Massachusetts
Contract No. F19628-77-C-0001

REVIEW AND APPROVAL

This technical report has been reviewed and is approved for publication.

Jack Segal

JACK SEGAL, GS-14
Project Engineer

FOR THE COMMANDER

Ronald E. Byrne Jr.

RONALD E. BYRNE, JR., Colonel, USAF
Director, Advanced Planning
Deputy for Development Plan

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: ESD-TR-77-137	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): GEOGRAPHIC DATA DISPLAY IMPLEMENTATION	5. TYPE OF REPORT & PERIOD COVERED
	6. PERFORMING ORG. REPORT NUMBER: MTR-3349
7. AUTHOR(s): D. H. LEHMAN	8. CONTRACT OR GRANT NUMBER(s): F19628-77-C-0001
9. PERFORMING ORGANIZATION NAME AND ADDRESS: The MITRE Corporation, Box 208, Bedford, MA 01730	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: Project No. 7090
11. CONTROLLING OFFICE NAME AND ADDRESS: Deputy for Development Plan, Electronic Systems Division (AFSC), Hanscom Air Force Base, MA 01731	12. REPORT DATE: JUNE 1977
	13. NUMBER OF PAGES: 226
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report): UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

COMPUTER SOFTWARE
COMPUTER GRAPHICS
DATA BASE
DISTRIBUTED PROCESSING
GEOGRAPHY

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

To support automated displays of positional intelligence data, detailed geographic background displays are needed over a wide range of scales. The Geographic Data Display System (GDDS) displays geographic data on a raster scan display and allows the user to zoom and translate around a map of Central Europe. As the user zooms in on an area, the area is displayed in greater detail, and geographic features such

DD FORM 1473, 1 JAN 73 — EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. ABSTRACT (Concluded)

as rivers, roads, etc., are added to the display. This report outlines the capabilities and design of the GDDS, describes the implementation, and documents the programs.

ACKNOWLEDGMENTS

This report has been prepared by The MITRE Corporation under Project No. 7090. The contract is sponsored by the Electronic Systems Division, Air Force Systems Command, Hanscom Air Force Base, Massachusetts.

TABLE OF CONTENTS

		Page
LIST OF ILLUSTRATIONS		5
LIST OF TABLES		6
SECTION I	INTRODUCTION	7
SECTION II	GEOGRAPHIC DATA DISPLAY SYSTEM	10
	INTRODUCTION	10
	OPERATOR CONTROLS	10
	TRANSLATE AND ZOOM	10
	Translation	13
	Zoom	13
	FEATURE SELECTION	16
	Format	16
	Macro Expansion	16
	Menu Operation	18
	OPERATING MODES	19
	Automatic	19
	Normal	19
	Special	20
	SYSTEM USE	20
SECTION III	GEOGRAPHIC DATA DISPLAY DESIGN	22
	INTRODUCTION	22
	DATA BASE PREPARATION	22
	REQUIRED USER FUNCTIONS	24
	Scaling and Levels of Detail	26
	Translating and Neighborhoods	26
	DATA STRUCTURES	28
	Detail Levels and Blocks	28
	Translation with Blocks	30
	Zooming with Blocks	31
	DATA MANAGEMENT	31
	Data Organization	34
	Index Organization	34
	Retrieval Process	34
	PARALLEL DATA BASES	38
SECTION IV	IMPLEMENTATION TOOLS	41
	INTRODUCTION	41
	SYSTEM ARCHITECTURE	41
	Design Philosophy	41

TABLE OF CONTENTS (concluded)

		Page
	MESSAGE PROCESSOR	43
	GRAPHICS PACKAGE	44
	Instancing	45
	Coordinate Systems	45
	Use by GDD	46
	Implementation	48
	FILE MANAGEMENT PACKAGE	48
SECTION V	IMPLEMENTATION DESIGN OF THE GDD	49
	INTRODUCTION	49
	MODULE COMMUNICATION	49
	MENU MODULE	51
	MENUUP	53
	MESLCT	53
	FUNCTION REQUEST HANDLER	57
	Zoom	57
	Translate	59
	DATA EXCEPTION MODULE	59
	Module Entry	61
	Automatic Mode Zoom	61
	Normal Mode Zoom	61
	Special	62
	Translation	62
	DATA BASE MANAGEMENT	63
	MDISP	63
	RPCOL	65
	MODE	65
	INITIALIZATION	67
APPENDIX I	DATA BASE CONSTRUCTION PROGRAMS AND PROCEDURES	68
APPENDIX II	SYSTEM INITIALIZATION PROGRAM	79
APPENDIX III	GDD OPERATING PROCEDURES	102
APPENDIX IV	COMMONS	104
APPENDIX V	PROGRAM SUMMARY SHEETS	123
APPENDIX VI	PROGRAM LISTINGS	169

LIST OF ILLUSTRATIONS

Figure		Page
1	Display and Controls of the Geographic Data Display System	11
2	Layout of GDD Function Key Pad	12
3	Translation Before and After Sequence	14
4	Zoom Before and After Sequence	15
5	Menu Format	17
6	Detail Ranking Algorithm	25
7	Loss of Context Due to Lack of Detail	27
8	Translation with Neighborhoods	29
9	Translation with Blocks and Neighborhoods	32
10	Zooming with Blocks and Detail Levels	33
11	Blocking and Ordering a Detail Level	35
12	Ordering and Format of an Index for a Detail Level	36
13	Data Base Retrieval Process	37
14	Parallel Data Base Selection	39
15	Schematic of 7090 Computer Facility	42
16	Inclusion of One Image into Another	47
17	Internal Communications of the Six GDD Modules	50
18	Flowchart of MENUUP of Menu Module	54
19	Flowchart of MESLCT of Menu Module	55
20	Flowchart of Function Request Handler	58
21	Flowchart of Data Exception Module	60
22	Flowchart of MDISP Entry into Data Management Module	64
23	Flowchart of RPCOL Entry into Data Management Module	66
24	Generated Entry and Exit Points for Crossed Blocks	70
25	Assignment of Single-Point Chains	70
26	Example Teletype Input for IMAGE Program	76
27	IMAGE Program Output Restating Input Parameters	77
28	IMAGE Output Showing Number of Chains and Points per Map Block, Drum Block Address and Byte Entry in Drum Block for Each Block of Data	77
29	Output of IMAGE Program Showing Index for Column of Data	78

LIST OF TABLES

Table		Page
I	Module Subroutines	52
II	Card #1	84
III	Card #2	85
IV	Card #3	86
V	Card #4	87
VI	Card #5	88
VII	Card #6	89
VIII	Card #7	90
IX	Card #8	91
X	Card #9	92
XI	Card #10	93
XII	Card #11	94
XIII	Card #12	95
XIV	Card #13	96
XV	Card #14	97
XVI	Card #15	98
XVII	Card #16	99
XVIII	Card #17	100
XIX	Card #18	101
XX	COMMUN COMMON	107
XXI	CURSTA COMMON	108
XXII	DATSTA COMMON	110
XXIII	ERASE COMMON	112
XXIV	COLORS COMMON	113
XXV	DATBAS COMMON	114
XXVI	FAC COMMON	116
XXVII	FILE COMMON	117
XXVIII	MACRO COMMON	118
XXIX	MAP COMMON	119
XXX	MENCON COMMON	120
XXXI	MNUTIA COMMON	121
XXXII	TREES COMMON	122

SECTION I

INTRODUCTION

Geographic Data Display (GDD) is a tool developed by Project 7090 to aid the real-time display of operational and intelligence information. This data is readily available in Command, Control and Communication Centers, but it is available in such copious quantities that it must be summarized and properly displayed to be useful. This "properly displayed" is the motivation behind the GDD.

Major subsets of operational and intelligence data are positional in nature, and therefore any method of representing the data for quick reference by an operator requires a map. For a more detailed analysis the actual raw data of latitude, longitude points must also be available to an operator, but obviously an operator can more rapidly assimilate information from a graphics image than from a list of data points.

One intention of Project 7090 is to develop techniques for displaying operations/intelligence information over a wide range of granularity — from data summarized over a large area of several hundred miles to individual reports displayed over only ten square miles. To support this summarization task, maps are required that adequately represent the geography at any needed scale. The Geographic Data Display System has the needed capability to project detailed maps as background for information displays over a wide range of scales.

The specific problem addressed by the GDD is, then, the manipulation of geographic data to provide adequate resolution of geographic features over a wide range of scales. If more data is displayed than can be absorbed by the resolution of the display device, the features will appear fuzzy. If too little data is used, the geography will appear sparse and angular, and the viewer may lose any sense of context. Since the purpose of the GDD is to allow a viewer to zoom in and out on a displayed map and still maintain clear, detailed geographic feature representation, an ability to dynamically vary the amount of data used for the display had to be developed. The solution to this problem used by the GDD was levels of detail.

A level of detail of a geographic feature is a map containing a fixed amount of data representing that feature. It can only be displayed over a relatively small range of scale before it gives fuzzy or angular displays. For any feature several levels of detail are defined; each successive level contains more data than the last, and each level is displayed only over its defined scale range. When the user zooms out of the scale range of a level of detail, the display is defined from the next level of detail. Such a scheme solves the problem neatly and puts no limit on the scales that can be displayed by the system.

To make the GDD even more flexible, not only is the amount of data displayed variable, but the actual geographic features displayed can be varied by the user to tailor the display to his needs. There is a feature library in the system containing coast lines, political boundaries, rivers, roads, etc. The user can select from this library the features he wants displayed for his particular application. Each individual feature is divided into levels of detail, allowing the detail of features to be adjusted independently of one another.

Another paper, ESD-TR-77-360, "Geographic Data Base Development," thoroughly describes the data base preparation process and programs. This paper, then, is intended primarily as implementation level documentation for the GDD. It will, however, give the reader a broad overview of the system.

The first section simply describes the GDD - how the user sees it and what it does for him. The next section is an overview of the conceptual design which begins by establishing the user needs and describing the data bases the system will use. The section then develops the design of the GDD around these givens and concludes with the data structures and data management techniques used in implementation. Section IV covers the system architecture and software tools used to implement the GDD. The final section provides top level documentation for each of the six modules of the GDD. The appendices contain additional, more detailed documentation of the programs, variables and operating procedures. The appendices assume a working knowledge of the 7090 computer facility's operating system.

SECTION II

GEOGRAPHIC DATA DISPLAY SYSTEM

INTRODUCTION

To the user the GDD is a TV screen on which maps are projected as background for operation and intelligence information displays. Using a trackball and function keyboard the user can invoke a few basic functions for manipulating the display and tailoring the display to his needs. In this section, these functions and other capabilities of the GDD are described as they appear to the user.

OPERATOR CONTROLS

To operate the GDD, the user sits in front of a TV screen with a function keyboard and a trackball positioned near by. Figure 1 shows an operator working with the GDD.

The function keyboard diagrammed in Figure 2 has twelve buttons, of which eight are currently used. Three of these are for zoom and translate requests, two for feature selection from a menu and three for changing the modes of the system. These functions are described below.

The trackball controls the position of a cursor on the TV screen. The cursor is used to select a point on the display for use in performing a translate, zoom or feature selection function.

TRANSLATE AND ZOOM

The portion of the world displayed on the TV screen can be fully described by its center point and extent*. The user can

*Extent is defined as the inverse of scale. The GDD was implemented using variables representing extent. To be consistent, extent is used throughout the document.

Figure 1. Display and Controls of the Geographic Data Display System

	ZOOM OUT	ZOOM IN
TRANS-LATE		
MENU	SELECT	
AUTOMATIC	NORMAL	SPECIAL

IA-49,343

Figure 2 LAYOUT OF GDD FUNCTION KEY PAD

manipulate the display he sees by manipulating the center point and extent with translate and zoom functions.

Translation

Translation changes the center point of the displayed map. The user selects a point on the display screen by positioning the trackball-controlled cursor over the desired point. When the translate function key is hit, the GDD moves the point designated by the cursor to the center of the display screen. The photographs in Figure 3 show a before and after sequence of a translate. Note that part of the map that was not in the original display has been brought on from off the screen. In essence, the user is viewing the map through a restricted window. As the user translates, he moves the window around the map to view a different area. If the user translates out of the mapped region, he will see a boundary line marking the edge of the map. Beyond this edge the map will be blank.

Zoom

With the zoom function the user can alter the extent of the area displayed around the center point, effectively changing the size of the restricted window in the analogy used above. The operator uses the trackball to position the cursor over the point he wishes to remain stationary during the zoom. The user hits either the zoom in or zoom out function button causing the distance between the selected point and all other points in the display to be either multiplied or divided by the magnification factor (normally 1.5). The result is a change of extent around the cursor as seen in Figure 4, another before and after sequence.

This is the simplest zoom that is done by the system. In Section I we spoke of displaying maps of adequate detail at all scales. This adjustment of detail is performed automatically by the GDD whenever a zoom is requested by the system. Thus, after

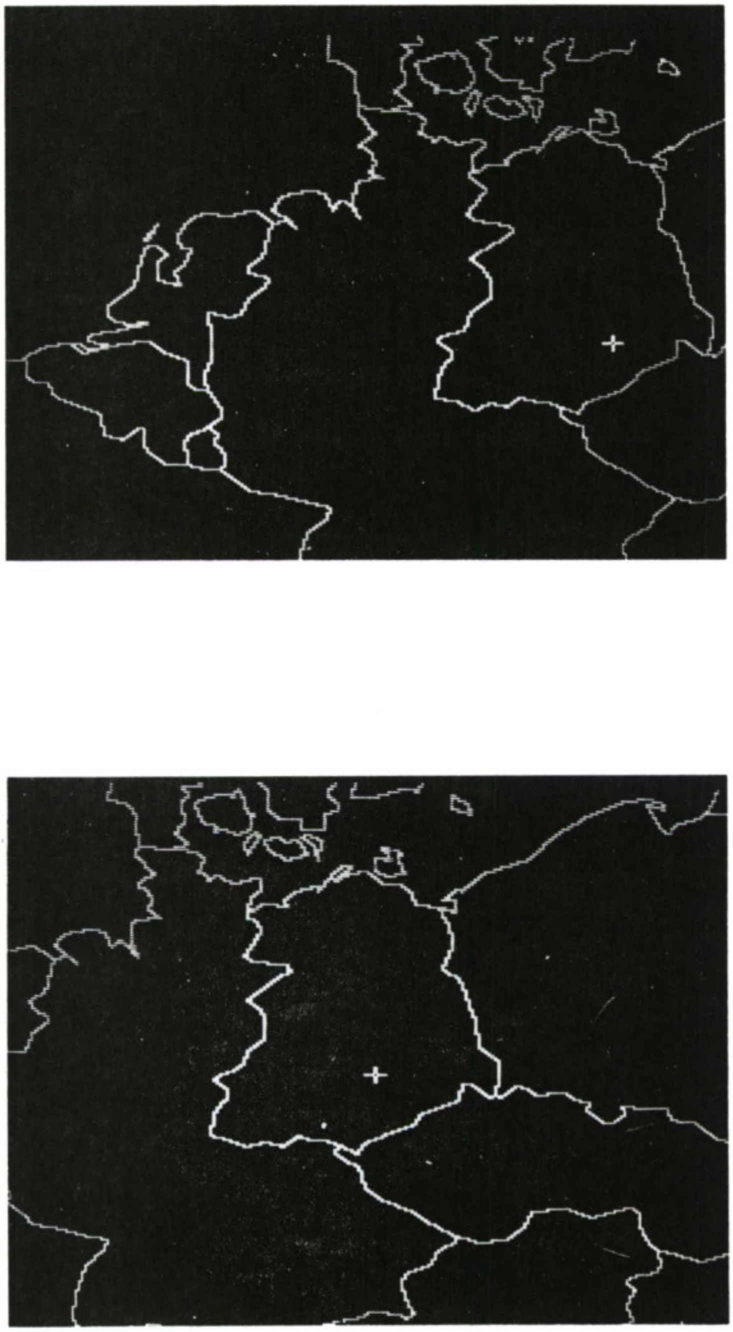


Figure 3. Translation Before and After Sequence

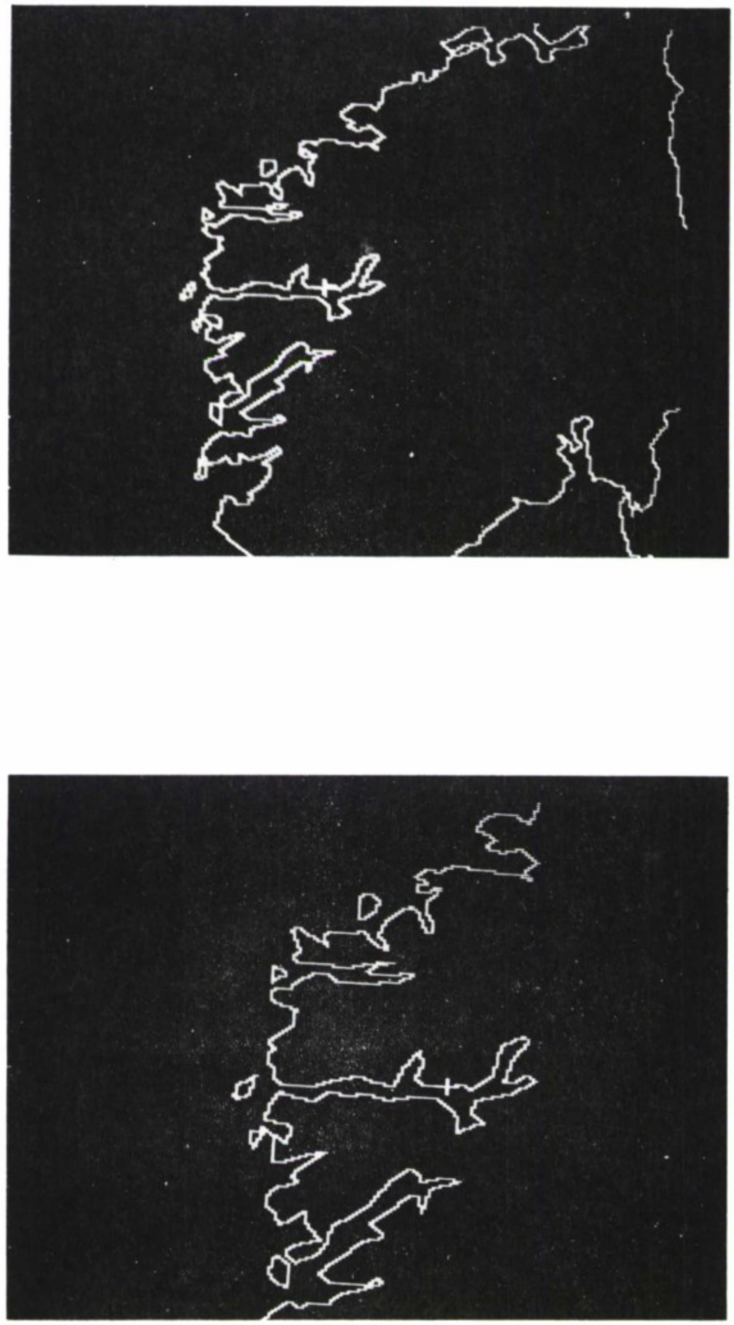


Figure 4. Zoom Before and After Sequence

the zoom described above is done, the system determines if there is too much or too little data displayed on the screen for the current extent value and adjusts the amount of data accordingly.

FEATURE SELECTION

In the introduction we also spoke of the operator having the ability to tailor the display to his own needs. By this we mean that the operator controls what geographic features are displayed on the screen. These features include political boundaries, roads, rivers, railroads, etc. In order to have an effective display, the user must be able to turn these features on and off. In the GDD this is done with a menu.

Format

The menu format is shown in Figure 5 and is displayed on the screen when the "menu" function button is pressed. The menu presents the user with a list of features available for display; those that are currently displayed contain a number in the ON column indicating the amount of detail with which the feature is currently displayed relative to the total amount of data available for that feature. A low number indicates little detail, a higher number more detail. The number in parenthesis directly opposite the features in the list indicates the maximum amount of detail available. The last line on the screen indicates which of three operating modes the system is in. These modes are discussed in the next subsection.

Macro Expansion

In the system there is a macro expansion capability for feature selection. A feature in the menu list may represent several feature data bases. For example, the list could contain the word "boundaries," which, when selected, would be expanded to represent the two data bases - coastline and political boundaries, each of which could

MAP FEATURE SELECTION		
FEATURE (MAX)	ON	OFF
MAP (3)	2	
RIVERS (2)	I	
ROADS (2)		
NORMAL MODE	MAKE SELECTION	

IA-49,344

Figure 5 MENU FORMAT

also appear separately in the list.

Menu Operation

The user picks a feature for display or deletion by positioning the cursor opposite the proper feature and beneath either the ON or OFF column and hitting the "select" function key. The system will immediately respond with an acknowledgement. The response could be an error message displayed at the bottom of the screen. Possible error conditions are the cursor is not positioned opposite one of the features or beneath either of the functions or a feature is selected for deletion which is not currently displayed. If the selection request is allowed, a message is displayed at the bottom of the screen and either an X is displayed in the OFF column if a delete was requested or a number representing the relative amount of detail with which the feature will be displayed is inserted in the ON column. The amount of detail with which a selected feature will be displayed is determined by the system as a function of the current extent of the display window.

Before any of the selected data bases are displayed or erased, the user can correct any of his choices. If an operator has selected a feature for display and then decides he no longer wants it displayed, he can select that feature again in the OFF column and the effect of the previous selection in the ON column is nullified. The same is true if the user has inadvertently chosen to delete a feature. Selecting it again in the ON column will produce no effect on the display of that feature.

Once the user is satisfied with his selection of features, he hits the "menu" function key again. This enters the user's choices and erases the menu from the display screen. Those features deleted by the user will disappear from the screen, followed by the addition of any newly selected features.

OPERATING MODES

The GDD will operate in three different modes - automatic, normal and special. These modes control the degree to which the user can adjust the amount of data on the screen. Only feature selection and levels of detail are effected by the mode change.

Mode changes are caused by hitting one of the three function buttons shown at the bottom of Figure 2. The system is initialized in automatic mode and remains in that mode until a mode key is pressed. The system stays in a mode until another mode is selected by the user.

Automatic

Automatic mode allows the user the least flexibility of the three modes. The user has no control over what features are displayed. The user can simply zoom and translate. As he zooms in (or out), features are displayed (or deleted) as predetermined extent thresholds for each feature are crossed. The amount of data for each displayed feature is also automatically adjusted to the extent. If a menu is requested, it is displayed, but only as a status report on what features are currently displayed; no feature selection is allowed in automatic mode.

Normal

Normal operation allows the user to select and delete features. When the user shifts from automatic or special mode to normal mode, the features that are currently displayed become the selected features. By using the menu to display or delete features, the user can change this list of selected features to adapt the display to his task. As in automatic mode, the amount of detail shown for a particular feature is a function only of the extent of the displayed map. Once a feature is selected for display, it will not disappear from the screen until the mode is changed; it is deleted by a menu request or the extent thresholds for that feature are exceeded. In the latter case, the feature will be redisplayed as soon as the extent is again within the maximum and minimum thresholds for the feature.

Special

Special mode is similar to normal mode in all but one respect. It locks the displayed features at their current levels of detail. No matter how much zooming is done, the amount of data displayed remains constant. If a feature is selected via the menu in special mode, the feature is displayed with detail appropriate to the current extent. However, as long as the system remains in special mode, no amount of zooming will alter the amount of detail displayed.

If the user shifts from special to either automatic or normal, detail levels of the currently displayed features are adjusted on the next zoom or translate request. In the case of a shift to automatic or normal modes, entire data bases may be deleted if the extent after the zoom is not within the predetermined thresholds for that feature.

SYSTEM USE

Fine, the user can control the system and look all around a map at various scales. How is the map useful to him?

The answer to this question is that the GDD is not the only process running on the computer as the operator zooms and translates. When the user requests a zoom or a translate, the GDD responds by relocating or scaling the map, but the GDD is not the only process to receive these function requests. Processes controlling foreground displays of, for example, radar or intelligence data, also translate or scale the displayed foreground data under their control.

These processes, running independently of the GDD and requiring only the center point and scale of the displayed area, are now being developed under Project 7090. This task of 7090 is investigating methods of summarizing and displaying the large volume of operational and intelligence data available to a C^3 operator. The map provides a background on which large quantities of data can be summarized and rapidly assimilated by a viewer. The location of the data summarized on the map is, of course, controlled by the center point chosen by the operator. The degree of summarization will be a function of extent. As the operator zooms in on an area, the foreground displays will provide more specific information.

SECTION III

GEOGRAPHIC DATA DISPLAY DESIGN

INTRODUCTION

This section is intended to be a brief overview of the Geographic Data Display System design. The first subsection describes the initial geographic data base and how it was prepared for use by the GDD. With the data base as a given, the needed user functions are specified in the next subsection. The following two subsections discuss how the data base must be structured and managed to implement the user functions.

DATA BASE PREPARATION

The original data base used for this project was World Data Bank I obtained through the National Technical Information Service. The data base contains approximately 80,000 latitude/longitude points in both degrees and radians, outlining the coastline and political boundaries of the world. For data management purposes the data base is divided into entities called chains. A chain is defined as a set of points which, when connected in order, form part of a geographic boundary. The chains vary in size from one point for small islands to several hundred for an intricate portion of coastline, such as the Norwegian fiords.

The chain format is the standard format for all geographic features in the GDD feature library which have linear characteristics.

The data massaging process described below was applied to all data bases used by the GDD, but the coastlines and boundaries of World Data Bank I are used as an example.

Only a short outline of the massaging given to World Data Bank I and the other data bases will be presented. ESD-TR-76-360, "Geographic Data Base Development," discusses the entire data base preparation problem and documents the programs needed for this process.

First, to obtain a data base of more manageable size, a subset of World Data Bank I was made consisting of those points falling within the region bounded by the points (63 N,033 W), (68 N,039 E), (30 N,012 W), (32 N,021 E). This is (roughly) Europe from Iceland to Moscow, Algeria to North Cape, Norway. All discussions that follow deal only with this European subset consisting of approximately 8200 points.

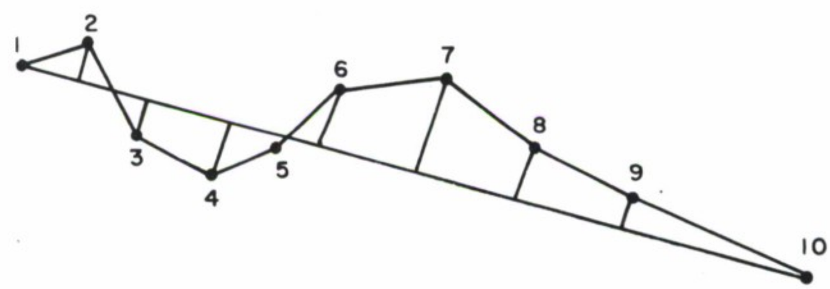
These latitude/longitude points of Europe were projected into an X,Y plane to form a map. The projection used was a Secant Conic with two standard parallels to minimize distortion. Scale error is 0% along the standard parallels (57 N and 41 N) and a maximum of 1% on the extremes of the map.

Once the subset was established and projected, the individual points were ranked according to their importance to map detail. This was necessary since displaying all 8200 points simultaneously results in wide fuzzy boundary lines on the display screen. At a scale which enables the entire region of the map to be seen, many points in the data base are too close to be resolved by the display into distinguishable points causing fuzziness. A program was written using an algorithm developed by the Harvard Laboratory for Geographic Display and Spatial Analysis which ranks the points according to their importance relative to a visible feature. Very simply, the trend line length between the endpoints of a chain in the data base is calculated. The distance of each point in the chain from the trend line is calculated and compared to a set of tolerances. Those points within a small tolerance of the trend line itself are considered least important since they describe a very small feature and are assigned to a low rank or detail level. The points falling farthest away from the trend line are considered the most important as they describe a gross geographic feature and are assigned to a high detail level. Figure 6 shows the process graphically. ESD-TR-76-360 describes the algorithm in detail.

A ranking of points into these detail levels provides the capability of displaying maps while controlling the detail and the resolution by varying the detail level of the points displayed. The detail level chosen controls the number of points displayed. For displaying large areas only important points are used; if a small area is displayed points of a lesser rank are also displayed. Thus, as the scale changes, approximately the same number of points are always displayed on the screen, but the points represent either more detail if a small area is being viewed, or less detail and more boundaries if a larger area is being viewed.

REQUIRED USER FUNCTIONS

The map display system had to have four functions - location, translate, zoom, and feature selection. Location is defining what part of the world is displayed. At the present this is an unimplemented function; the map is located in central Europe and cannot be varied. The zoom had to provide adequate detail in the geography for a wide continuous range of extent values. Furthermore, all three functions had to perform quickly to be useful in a real-time application. For the moment consider only zoom and translation of a single data base; speed and feature selection from



NOTE :—

DISTANCE FROM THE TREND LINE DETERMINES IMPORTANCE OF A POINT TO DETAIL. POINTS 2 AND 9 ARE LEAST IMPORTANT AS THEY DEVIATE LEAST FROM THE TREND LINE. POINT 7 IS THE MOST IMPORTANT SINCE IT IS FARTHEST AWAY AND THUS DEFINES A LARGER FEATURE THAN THE OTHER POINTS.

IA-49,345

Figure 6 DETAIL RANKING ALGORITHM

the feature library will be covered in a later subsection as generalizations of how a single data base works.

Scaling and Levels of Detail

Any good graphic display system has a scaling function enabling the user to magnify his displayed image. Consider the effect of this standard system on a map covering a large area and therefore containing only important points. As the map is magnified less area is seen, but boundaries become more angular and accuracy of representation is lost as the distance on the screen between points becomes greater and greater: curves would become sharp angles. Figure 7 shows a portion of Scandanavia after several zooms without the addition of detail. To maintain a recognizable image of Scandanavia a dynamic system would have to obtain more points from the data base as the map was magnified. Conversely it would have to delete points as the map was scaled in the opposite direction.

A compromise between the standard graphics system and a truly dynamic system was developed. In the GDD simple magnification is used over a specified extent range. When scaling is requested outside the range specified, the displayed map is replaced by a new map containing points of a higher or lower detail level, whichever is appropriate to the zoom direction. This new map is then scaled by the user until an extent threshold is crossed causing a new map composed of a new detail level to be displayed. Such a system provides resolvable detail at all practical magnifications since a map composed of points of a given detail level is only displayed over the range of magnification which it can support with adequate resolution.

Translating and Neighborhoods

The other primary design consideration was a translate function. This feature enables the user to move any point of the map currently

Figure 7. Loss of Context Due to Lack of Detail

visible to the center of the display screen. This, too, is a standard graphics tool, but the map posed an interesting problem. If a point on the extreme edge of the screen is translated to the center, half of the screen is left blank, unless points that are not in the current display are kept in memory, ready for instant display when a translate is done. This was impossible because of a memory size limitation. The entire map cannot be kept in instant readiness without using a considerable amount of memory. Again a compromise was reached.

Given a center point of the map and an extent range, a neighborhood around the center point can be defined larger than the maximum allowed extent such that the entire neighborhood will never be displayed as long as the extent range is not exceeded. The neighborhood is the shaded area in Figure 8a. As the map is translated within that neighborhood, undisplayed points in memory are displayed, and displayed points are dropped from the screen as shown in part b of Figure 8. When a user translates too close to an edge of a neighborhood, data no longer displayed and farthest from the displayed map is erased and new data bordering the displayed edge is brought in from secondary storage redefining the neighborhood as shown in Figure 8c. This system allows the user to have an instant translate without the inconvenience of a momentary blank screen.

DATA STRUCTURES

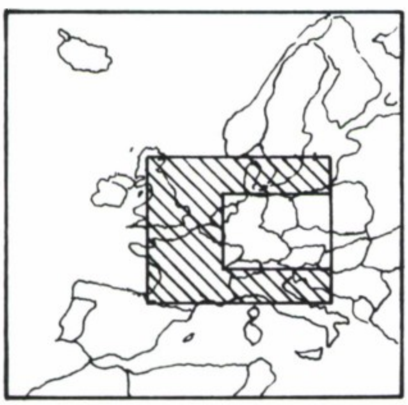
With the design considerations outlined above, the problem of building a geographic display system to give the needed fast responses reduced to a data structure and data management problem.

Detail Levels and Blocks

To implement the zoom function three complete maps of different levels of detail were constructed. The first map contains only

MAP REGION

a) DISPLAY WINDOW AND NEIGHBORHOOD



b) DISPLAY WINDOW AND NEIGHBORHOOD AFTER IMMEDIATE RESPONSE TO TRANSLATE

c) DISPLAY WINDOW AND NEIGHBORHOOD AFTER NEIGHBORHOOD HAS BEEN REDEFINED

IA – 49,346

Figure 8 TRANSLATION WITH NEIGHBORHOODS

the 800 most important points of Europe in World Data Bank I for use with the largest extent, displaying the entire European map which is approximately 2200 miles across. The 800 point figure is a practical restriction imposed by the memory requirements of the graphics system used (this graphics system, PALLET, is described in the next section). The second and third maps contain 2300 and the entire 8200 points, respectively. It is neither possible nor desirable to display the entire map with these last two large data bases. Only part of these data bases can be resident in core at any one time. That part not in the current neighborhood must reside in secondary storage until the viewing window is translated near to the edge of the data in memory at which time the neighborhood is redefined.

A structure was imposed on each of the three maps to enable a neighborhood around a center point to be selected. Each map was divided into square blocks. The first map into 9 blocks, the second into 81 and the third into 729. A block in the first data base was divided into 9 blocks in the second data base, and a block in the second was represented by another 9 in the third.

Translation with Blocks

With such a structure it is only necessary to keep a maximum of 16 blocks around the center point of the display in memory. As the center point is translated, the required 16 blocks change but not all at once. The extent ranges over which each level is displayed prevent the user from seeing more than a three-block width or height at any one time. Thus, there is enough undisplayed data to fill most of the screen when a translate to an extreme boundary is done. (If magnification is such that less than a three-block width is visible, there will always be enough data to fill the screen when a translate is requested.) When a neighborhood needs to be redefined, blocks not displayed can be erased and new blocks read in from secondary storage. Figure 9 shows this process. This type of structure allows the viewer an immediate translate capability; the viewer should not be aware that a data exception has occurred necessitating references to secondary storage.

Zooming with Blocks

The zoom capability is also instantaneous, with one exception. Whenever a zoom is requested, the data currently displayed is magnified accordingly. If a scale threshold has been crossed requiring a new detail level an entire neighborhood of 16 blocks must be read from secondary storage. So, though the operator sees an instantaneous zoom, there is a delay before he sees a new level of detail. His operation is, however, not interrupted at any point.

Figure 10 shows the zoom function implemented with blocks and neighborhoods. In the diagram it can be seen that a new neighborhood is defined from the next detail level when a threshold is crossed. The blocks of the new higher detail level cover a smaller geographic area than the previous level but contain at least as much data.

DATA MANAGEMENT

We have now developed a data structure that provides the capabilities and flexibility we need. The question now becomes how to manage a data base with such a structure to provide a user with fast response when a translate or zoom is requested. Since it is impossible to give an instantaneous response when a data exception occurs on a translate or zoom, the user is given an instantaneous partial response by simply magnifying or translating the data available in the neighborhood. Any new data is displayed as fast as it can be retrieved from secondary store. The data base management scheme used by the GDD uses an indexing system

IA - 49,431

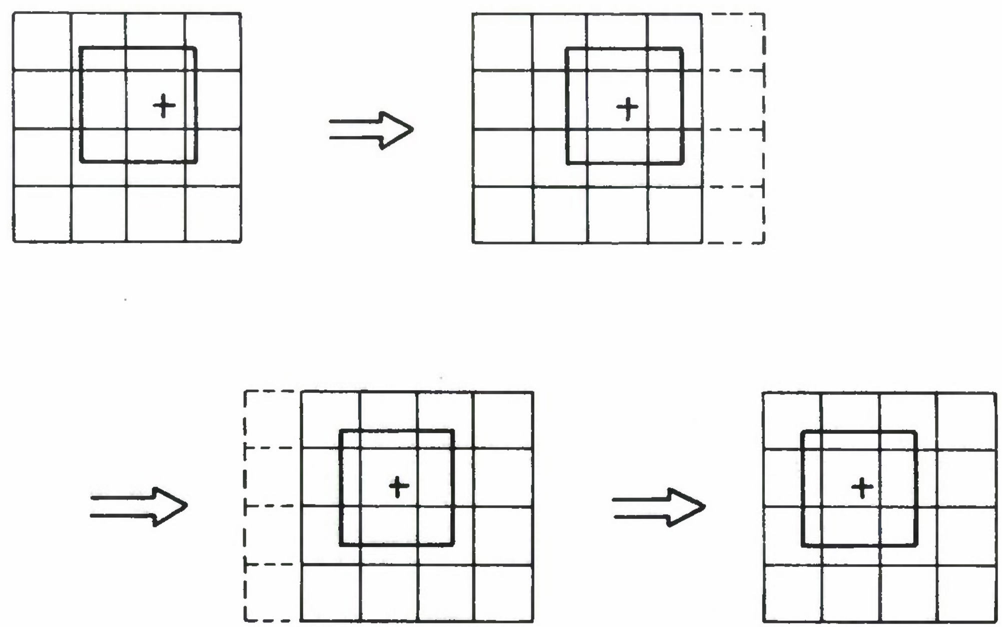
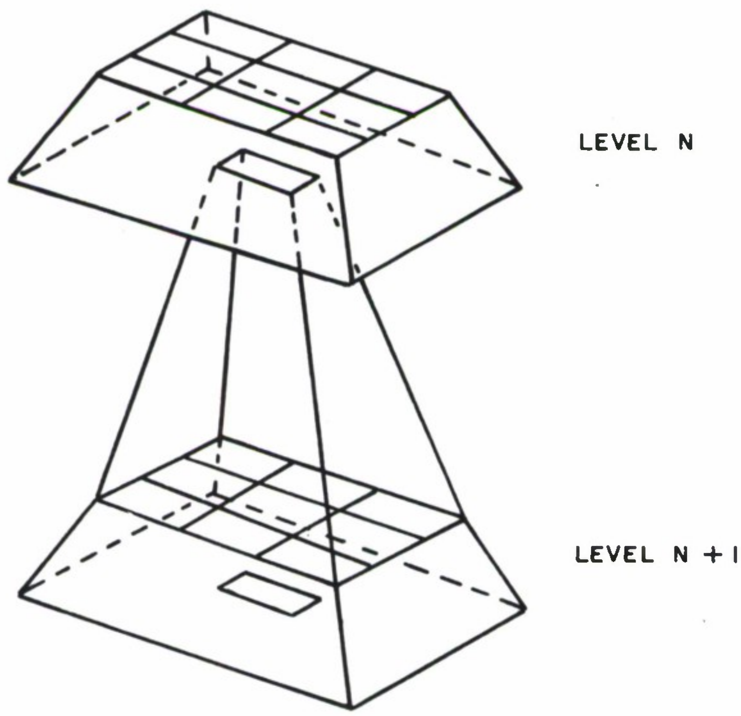


Figure 9 TRANSLATION WITH BLOCKS AND NEIGHBORHOODS



1A – 49,434

Figure 10 ZOOMING WITH BLOCKS AND DETAIL LEVELS which eliminates searches and minimizes the number of accesses to secondary store needed to perform this retrieval process.

Data Organization

To form the blocks with which neighborhoods are composed, a grid is superimposed over the map area. The geographic data for one detail level of a feature is placed into the block of the grid surrounding it. This blocking process is described in Appendix I. Once divided into blocks, the data is stored on secondary store in column order (this process is also described in Appendix I). This is shown in Figure 11 where blocks are stored in secondary storage contiguously in the order in which they are labeled. When the blocks are stored in this way, it is possible to retrieve a single four-block column of a 16-block neighborhood with one storage access since the four blocks are stored contiguously. This saves considerable time since it is the seek time, not the data transfer time, that causes the bottleneck in the retrieval process.

Index Organization

An index entry for each block is created and stored in row order in a file on the storage device. An index is diagrammed in Figure 12, where the index entries are stored contiguously in the order shown. Such an ordering allows the index entries of the four blocks which head the four columns of a neighborhood to be retrieved with a single storage access. An index entry contains two pieces of data - the storage address of the block and the length of contiguous data that must be read starting at this address to retrieve the data for all four blocks in the column of the neighborhood headed by the block.

Retrieval Process

Figure 13 puts the entire process together. Given a center point of the display, the intersection of the grid lines nearest

IA— 49,347

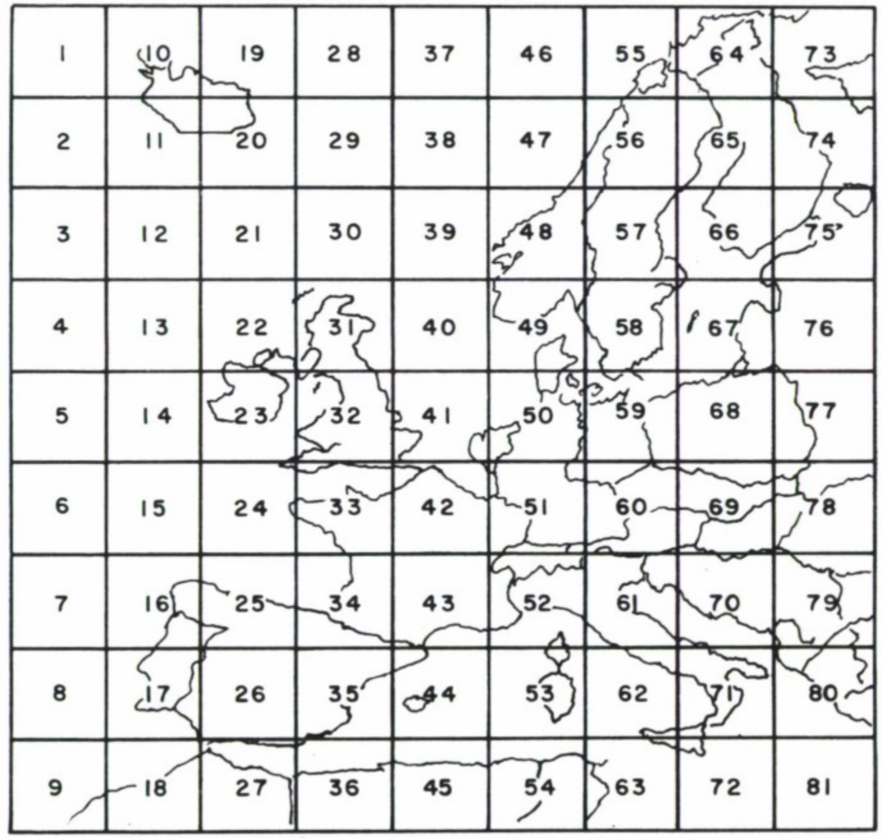
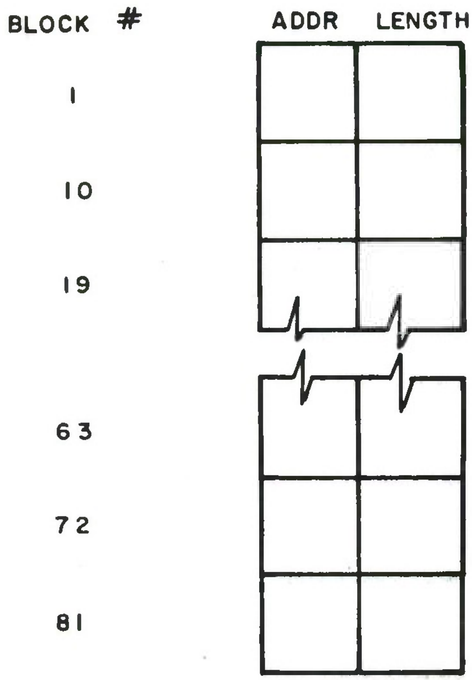


Figure 11 BLOCKING AND ORDERING A DETAIL LEVEL



1A-49,348

Figure 12 ORDERING AND FORMAT OF AN INDEX FOR A DETAIL LEVEL

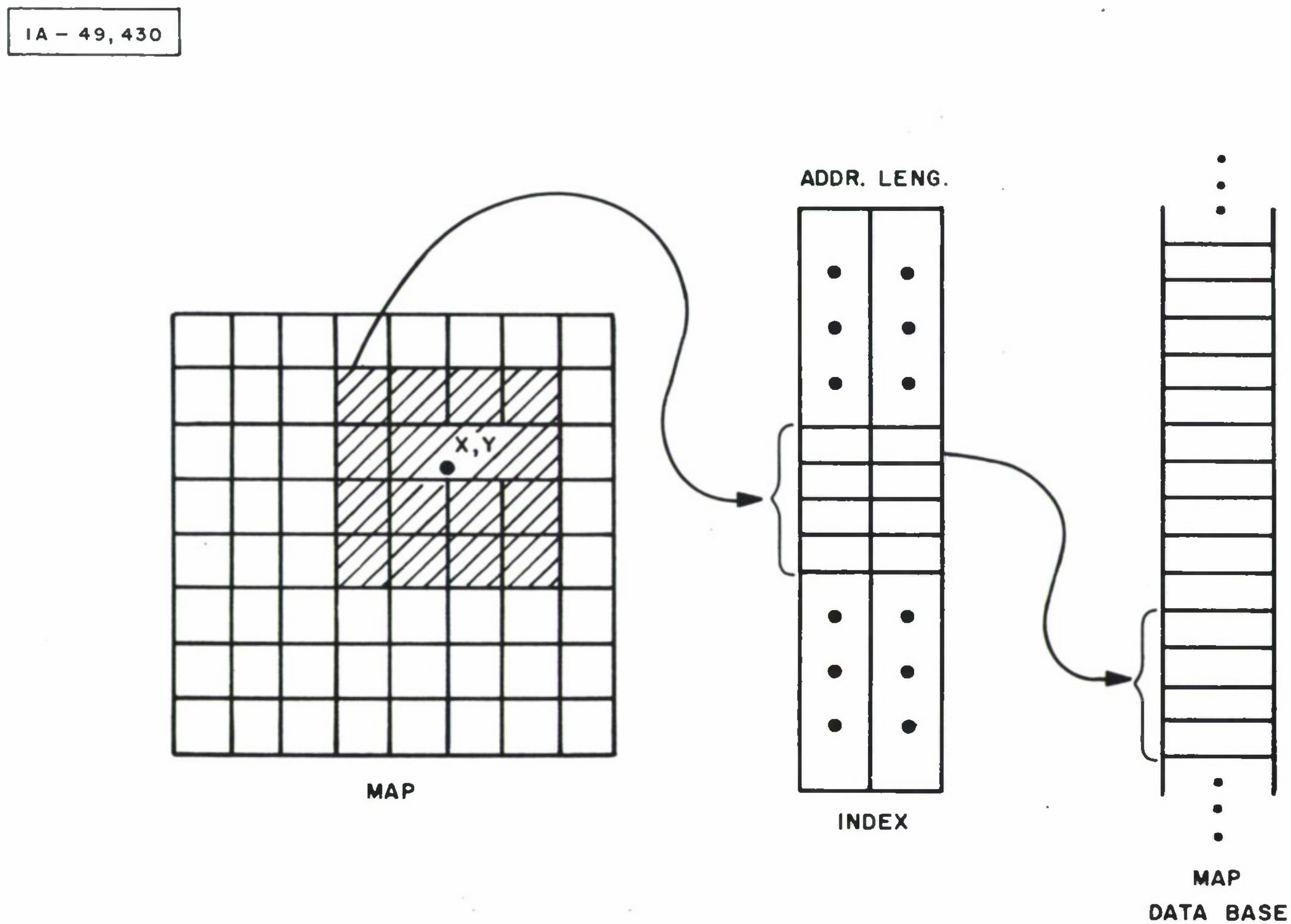


Figure 13 DATA BASE RETRIEVAL PROCESS

the center point can be determined. This grid point then determines the upper left block in the neighborhood. This block number is used as a pointer to the index to retrieve the four index entries for the four blocks heading the columns of the neighborhood. Since they are stored contiguously, one access is required. Using the length in the index, each column can be read from storage. Four accesses are required to retrieve the data for an entire neighborhood.

PARALLEL DATA BASES

The previous discussion of data management explains how a single detail level of a single feature is managed. The system does, however, have multiple features selectable by the user, and each feature has several detail levels. These multiple features are said to be handled in parallel with one another. That is, each feature in the feature library is treated as if it were the only feature data base available to the system. When a function is requested, each feature in the library is processed identically, one after the other.

The example in Figure 14 shows how parallel data bases are processed and how the correct detail level of each parallel data base is chosen. In Figure 14 three feature data bases are shown, each with several levels of detail. For each detail level, a range of extent over which that detail level provides adequate resolution is defined. (The ranges for detail levels of a feature overlap to prevent thrashing back and forth if a user zooms in and out around a threshold.) The vertical line in the figure shows the current extent of the display. The detail level of the features with which this line intersects is the one that should be displayed at this extent. It should be noted that there is no

IA – 49,432

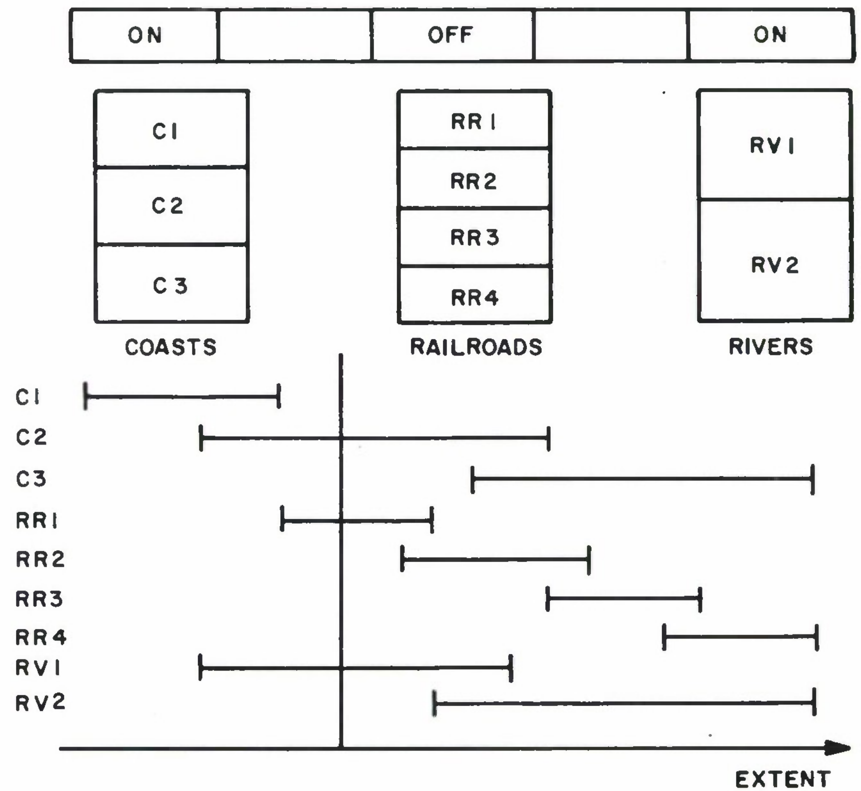


Figure 14 PARALLEL DATA BASE SELECTION

relationship between detail levels of different features. The fact that detail level 1 of one feature is displayed does not mean that another feature must be displayed at level 1, or, for that matter, displayed at all. Finally, before being displayed, a "user interest vector," shown along the top of the figure, is checked to see if the features allowed at this extent are wanted by the user. For those that are selected by the user, the file numbers of the data file and index file and the number of blocks into which the detail level has been divided are passed to the data management system. Thus, the data management system treats all detail levels and features the same; it simply retrieves the index entries and actual data from different files.

SECTION IV

IMPLEMENTATION TOOLS

INTRODUCTION

The GDD was implemented using 7090's existing distributed processing computer system. To support this system several large software packages were written: a message processor to support the distributed processing, a graphics display system, and a file management system. These programs were used to implement the GDD.

Below, the system architecture is presented followed by a brief summary of each of these software packages.

SYSTEM ARCHITECTURE

The 7090 computer facility is a two-computer distributed processing system. Figure 15 is a schematic of the system. The Interdata 70 (I-70) is the display processor driving a RAMTEK digital color television interface. It has 64K of memory and shares a 600-line-a-minute Data Products printer and a 200-card-per-minute card reader with the Interdata 4 (I-4). The I-4 with its 64K of core is the applications machine connected to a Vermont drum with a four-megabyte capacity. The two machines communicate with each other via a Bell 201 communications interface running at 25K baud. Both machines run the Interdata BOSS 4B operating system and can operate totally independent of one another. The actual display devices attached to the RAMTEK are a Conrac TV monitor and a large screen, ADVENT, projection TV. A trackball and function key pad are also connected to the RAMTEK.

Design Philosophy

The design philosophy of this architecture is summarized here. A large Command, Control and Communication system has many

IA-49,350

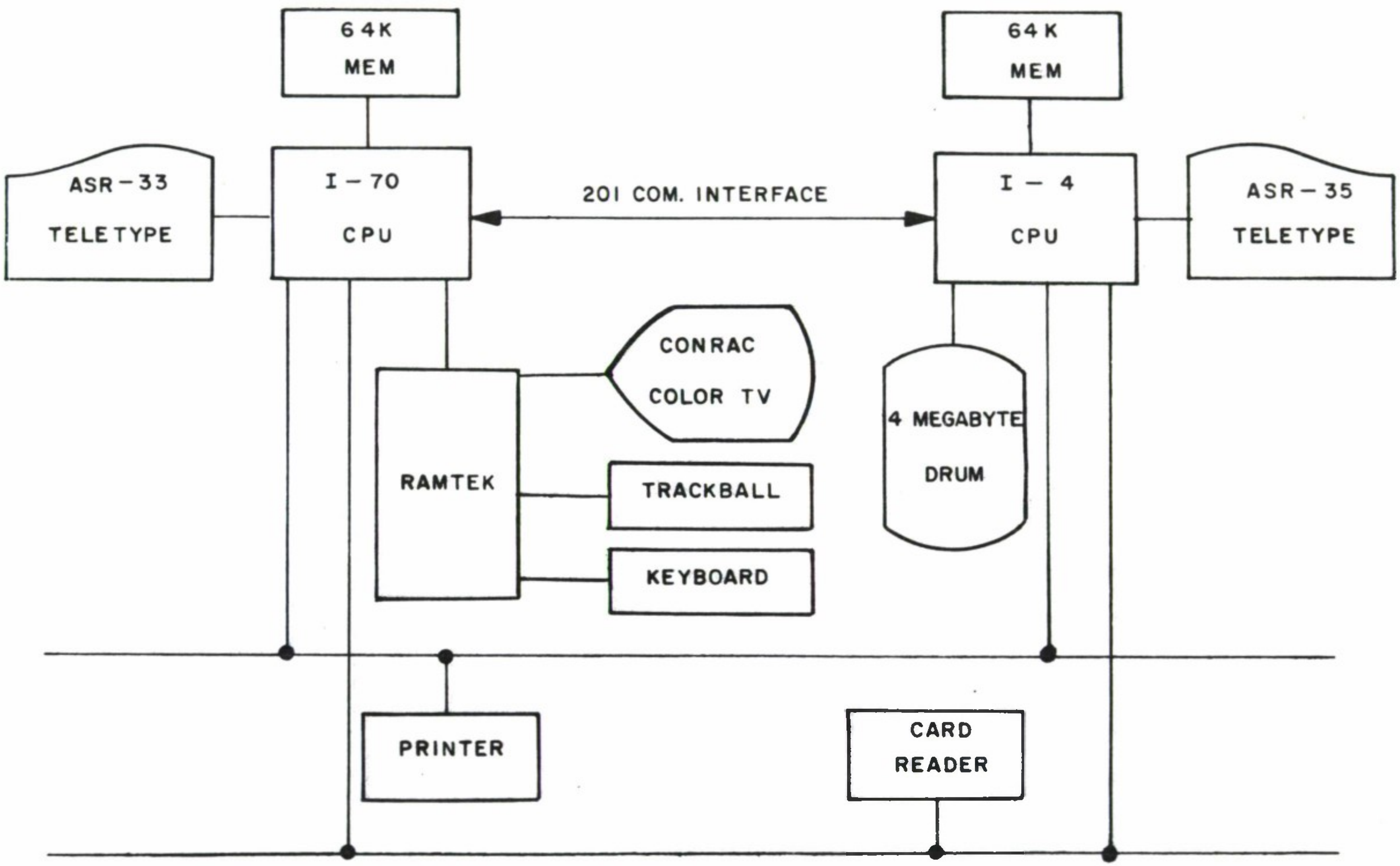


Figure 15 SCHEMATIC OF 7090 COMPUTER FACILITY

processes running concurrently with a need to communicate with each other. Since a process often needs to send a message simultaneously to more than one process, some of which are unknown to it, it makes sense to have a broadcast system that is receiver oriented. That is, a message is sent by putting it on a communications bus where it can be examined by each process running on the system. The process can either use the message or ignore it. The receiver decides what messages it wants, not the sender. Such a system cuts down on message traffic and relieves the application programmer of communications overhead.

In our system, the I-70 is a display processor running an operator display station. The I-4 is an application machine which handles the geography for the system. The data link substitutes for a bus, though the bus-receiver-oriented approach is simulated by the Message Processor, one of the software packages mentioned above.

MESSAGE PROCESSOR

The Message Processor program, MP, simulates the bus communication system discussed above. A copy of MP resides on each machine and acts as the system interface to the 201 communication device.

When an application program which will use MP is designed, the programmer must decide what information will be broadcast throughout the system by MP. This information is usually of global importance in nature, such as the center point and extent of the displayed map in the GDD. Once this is decided, a format for the messages containing information is formed and a type assigned to each message. The programmer can then write the different routines of the system, knowing that no matter how many routines want to receive a message, he only has to specify the type and send the message once. If a routine wants to receive a message of a certain type, the programmer must only make an entry in an MP table to that effect.

At initialization, the tables are constructed in MP identifying which programs want to receive what message types and on which machine each receiving program is resident. When a message is sent, the user calls MP with two arguments - the message and the message type. These two arguments are first sent to the MP on the other machine. The two MP's then check the message type against their internal tables. When a match is found in the table, MP invokes the program associated with that matched type. Each program is allowed to run to completion, at which time the next program in the table which wants to receive that message type is invoked. Each program thus appears to be continually examining a bus containing the message stream and picking off only the ones necessary for its operation.

It should be noted here that this implementation permits true distributed processing. There is no one large machine in control with several satellites; control is distributed between both machines, each doing its separate task independent of the other. It also should be noted that each machine is not dedicated to a single task; several processes are resident on each machine. The execution of these processes is controlled by MP as a function of the messages received and which process or processes want to receive that message. If several processes running on one machine degrade performance, the situation can be improved by adding another processor onto the bus. The theory of the bus operation puts no limit on the number of machines running in the network.

GRAPHICS PACKAGE

PALLET is a sophisticated graphics display program which provides the user interface to the RAMTEK digital TV driver. Through a series of subroutine calls the user can define images of points, lines, arcs, blocks of color and characters. Once defined, these images can be stored on drum or displayed on the RAMTEK. Any image stored on drum can be used along with points, lines, arcs, blocks and characters to form another image, which also may be displayed or stored. With the ability to refer to images or parts of images by name, change color, erase images and control cursor position and a function key pad, PALLET becomes a very versatile interface to the RAMTEK.

Instancing

PALLET is designed around the graphics concept of instancing. A graphics instance is a geometric form that can be used repeatedly, either in a single image or in many images. A common image is formed with lines, points, arcs, blocks and characters given a name and stored in secondary store. This image can now be an instance and used several times to form another image by recalling it by name. For example, the instance could be a representation of a window. In the construction of an image of a house, the instance of the window would be used several times, the only difference being the position of the window in the image of the house each time the instance was used.

Coordinate Systems

Position of the window in the house opens the Pandora's box of coordinate systems within PALLET. When an image is initially defined with an OPEN command, the coordinates of the lower left and upper right corners of the space for the image are given. This establishes the coordinate system of the display space for that image. When one of the primitive forms, lines, points, blocks or characters is placed in an image, it is positioned in the image according to the coordinate system with which the image was opened. If any of the X, Y coordinates of the points of the primitive form fall outside the display space, the forms are clipped off at the boundary. Now, when one includes an instance, that is, a previously defined image, into another image, one specifies where in the coordinate system of the new image the lower left and upper right corner of the display space of the instance should be placed. This nesting of the coordinate systems is shown in Figure 16.

Use by GDD

As an example of how PALLET coordinate systems work, let's look at how the GDD uses PALLET. PALLET is used by the GDD to display both the menu and the map. The menu is a straightforward application, declaring a display space and positioning characters within it. When the select function button is pressed, the cursor position is read. Since the position of items in the menu is known from when the image was constructed, the cursor position determines which feature and function have been selected. The map, on the other hand, is a bit more complex and more useful for tutorial purposes.

An image called "world" is opened from the lower left (0,0) to the upper right (511,479). (This coordinate system was chosen because the RAMTEK raster is 479 lines by 511 dots.) Into this image is included an image called "map." The display space of "map" is defined to be the corners of our European map given in the coordinate system of the projected map. The coordinates are given such that the corners of "map" fall within the "world" display space to give the proper initial center point and extent. If any of our map data which is in projected coordinates is now displayed in the "map" display space, it will appear in the correct position relative to any other piece of map data since the coordinate systems of the projected map and the display space of "map" are identical.

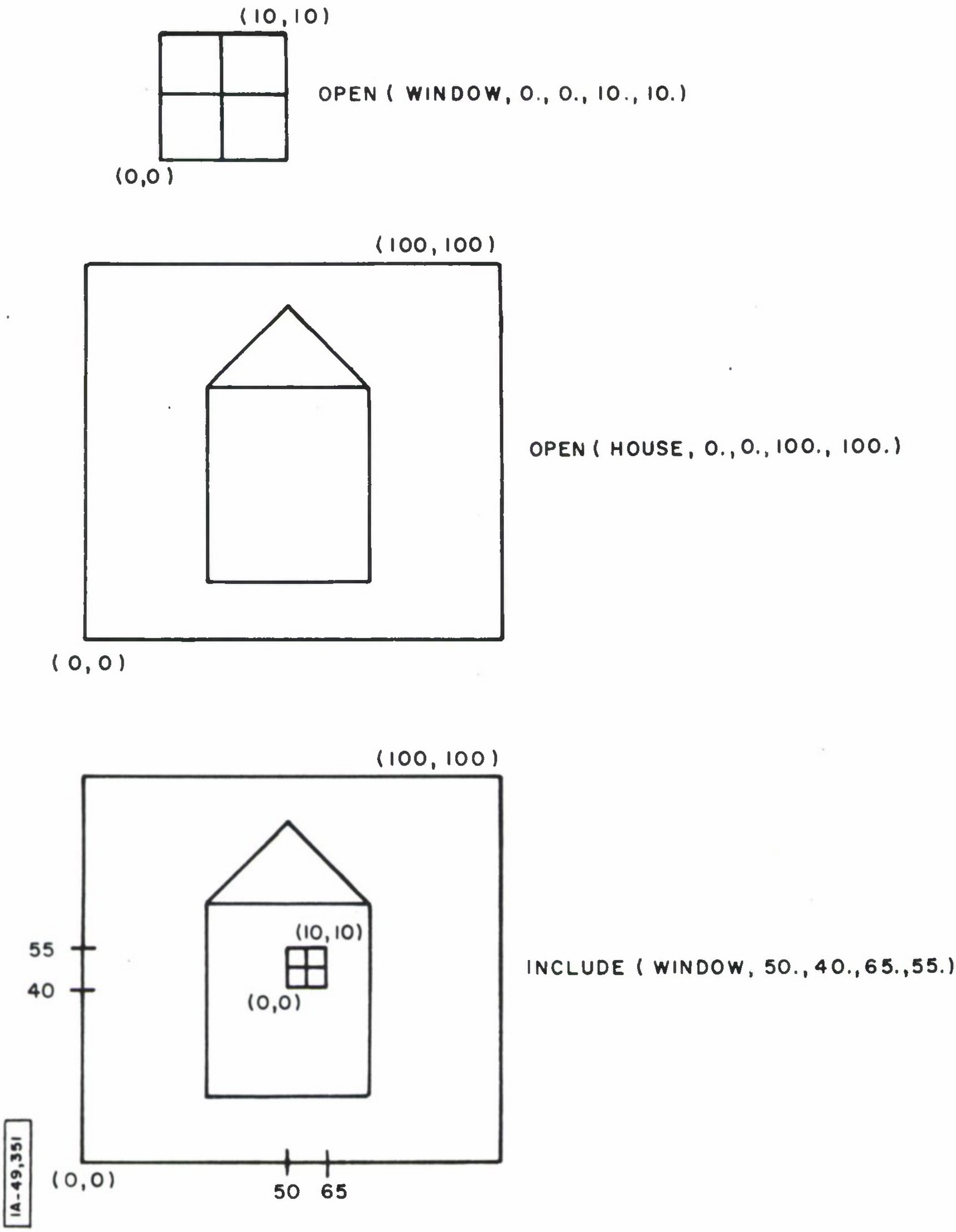


Figure 16 INCLUSION OF ONE IMAGE INTO ANOTHER

Implementation

PALLET has been implemented using both machines, a fact dictated by the system configuration. The part of PALLET controlling the display, interfacing directly with the RAMTEK, is resident on the I-70. Because of the use of secondary store for graphics instancing, that part of PALLET which constructs images is resident on the I-4. The two parts communicate via MP. Once an image is constructed and designated for display, it is sent out on the bus and received by the I-70 portion of PALLET, where it is processed and turned into a display list with the appropriate translation and extent applied to it.

FILE MANAGEMENT PACKAGE

The File Management Package (FMP) handles the storage and retrieval of data for PALLET. Through a system of subroutine calls to FMP, PALLET can store data in secondary store in a hierarchical file structure.

Except for opening the physical file used by FMP and initializing FMP when the GDD is executed, FMP is transparent to the GDD. No further description is necessary.

SECTION V

IMPLEMENTATION DESIGN OF THE GDD

INTRODUCTION

The Geographic Data Display program was designed as six separate modules written mostly in FORTRAN: Menu, Function Request Handler, Data Exception Handler, Data Base Management, Mode Change and Initialization. The Menu module allows the user to select and delete features with a menu. The Function Handler processes zoom and translate requests at the top level and passes the new center point and scale to the Data Exception module. This module then determines what, if any, detail levels and neighborhoods need changing. The Data Base Management module, written mostly in assembler, retrieves the new neighborhoods designated by the Data Exception module or the Menu module and passes the data to PALLET. The Mode module handles the mode change functions and Initialization sets the system for operation.

This section is designed to serve as top level program documentation. First, the communication between modules will be defined, and then the function and implementation of each module will be discussed separately. Individual subroutines are documented in Appendices V and VI.

MODULE COMMUNICATION

The six modules of the GDD communicate via common variables and MP. Within GDD, MP is used only between the Function Request Handler and the Data Exception Handler. This is necessary since the Function Request Handler has been implemented on the I-70 to enable faster response to the user. The other five modules all reside on the I-4 and only communicate with each other through the labeled common COMMUN. Figure 17 shows the interconnection of

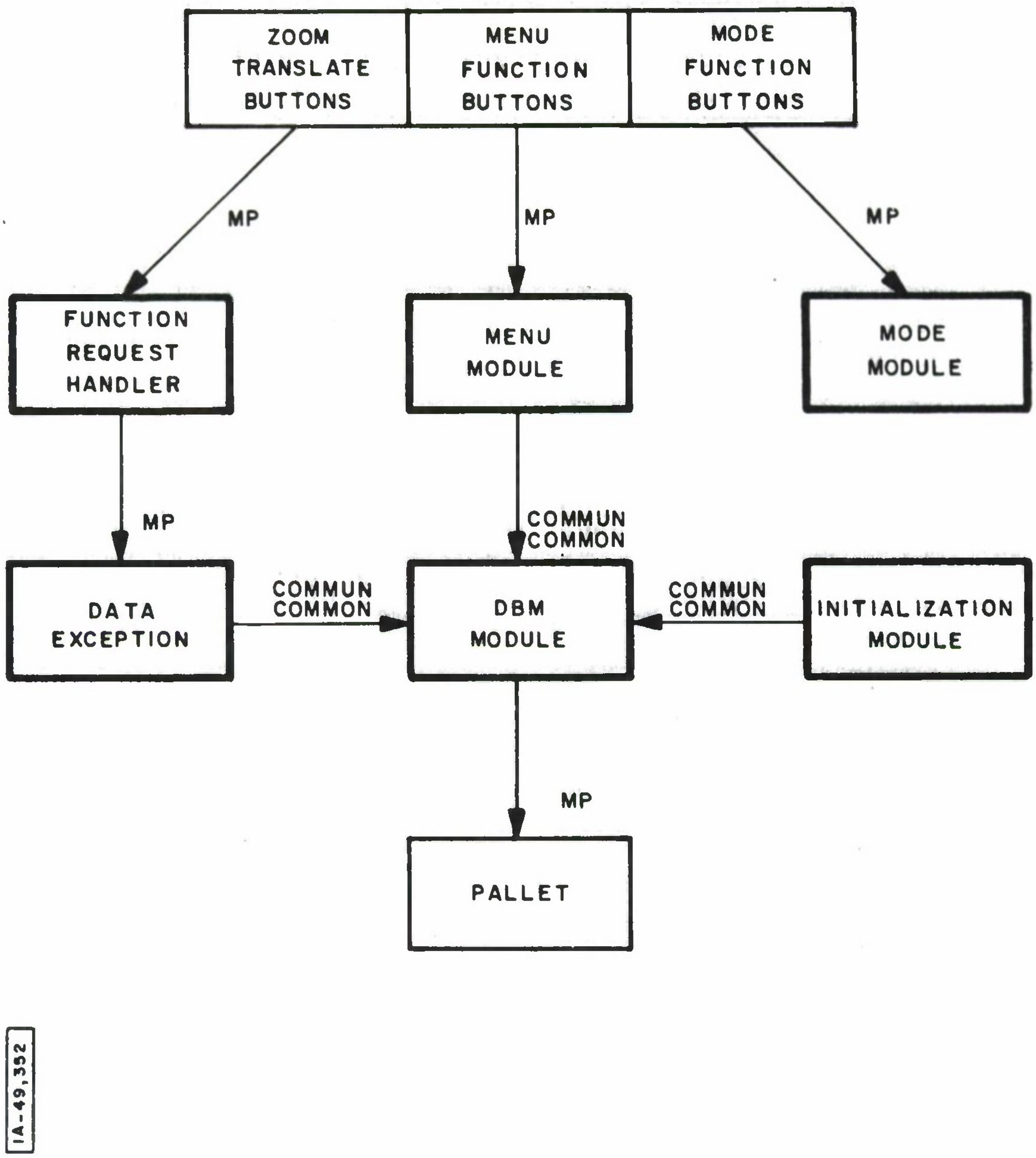


Figure 17 INTERNAL COMMUNICATIONS OF THE SIX GDD MODULES

the six GDD modules.

When a zoom or translate request is made the Function Request Handler broadcasts, via MP, the new center point and/or extent. The Data Exception module receives this message, stores the information in the CURSTA labeled common (all commons are documented in Appendix IV), and determines what new data need to be added and what data need to be deleted. These decisions are recorded in the SELECT and DELETE arrays of the COMMUN common. Each geographic feature available to the system has an entry in these two arrays. If a data base needs to be displayed or a detail level needs to be changed, the proper entry in the SELECT array is set equal to the detail level at which it should be displayed. If a currently displayed feature needs to be deleted the appropriate entry in the DELETE array is set non-zero. The relationship between entries in the arrays and feature data bases is discussed in Appendix IV.

The Menu module works in a similar manner. When the operator selects or deletes a feature, the proper entries in SELECT and DELETE are changed.

After either the Data Exception module or the Menu module have set COMMUN, the Data Base Management (DBM) module is activated by a subroutine call to MDISP. The DBM module then sends erase commands to PALLET for those features whose DELETE entry is non-zero and retrieves data from secondary store for those features whose SELECT entry is non-zero.

MENU MODULE

The Menu module processes all selection and deletion requests made via the menu. To do this, it makes heavy use of PALLET for displaying the menu and for responding to and receiving the user requests. Table I contains a list of the subroutines in the Menu

Table I

Module Subroutines

MENU	FUNCTION REQUEST HANDLER	DATA EXCEPTION	DATA BASE MANAGEMENT	MODE	INIT
CHARLV	CRTOMP	AUTOFZ	ALLOC,DEALOC	ATOFFS	IMPTAB
CLEVEL*	ERMSG	AUTONZ	CLMERS	AUTONS	INIT
CURPOS	NEWCEN	CLEVEL*	ERMSG	STATCS	REDCOM
DBPOS	SETMSG	CURSTA	GRDCEN*		STATIN*
MENUUP	STATIN*	GRDCEN*	INMVE		
MESLCT	TRANTP	MTRANS	MDISP		
NAME*	ZMINTP	ZMTRNS	MSEND		
RESPON	ZMOUTP		NAME*		
SETSTA	ZOMTOP		REDSND		
WRTCHR			RETREV		
			RINDEX		
			RPCOL		
			SETBF		
			SETINX		
			SETITM		
			TOPLFT		

* Subroutines shared by one or more modules.

module. MENUUP and MESLCT are two main line routines called when the proper function buttons are pressed.

The following discussion is in two parts. Under MENUUP the generation and display of the menu is discussed, followed by a description of the entry of the completed menu. Under MESLCT selection and deletion of a specific menu entry are discussed.

MENUUP

When the menu button on the function pad is pressed, the MENUUP subroutine is called to display the menu for use by the operator (refer to Figure 18). MENUUP first displays the basic menu stored in the PALLET file when the system is initialized (see Appendix II). It then creates an image called STATUS. MENUUP adds to the STATUS image the detail level of each of the features in the menu currently being displayed. A message stating the mode of the system is also included in STATUS before it is displayed using PALLET. If the mode is normal or special the variable MENU is set to allow the user to select and delete features, and MENUUP returns. If the mode is automatic, MENUUP simply returns since no feature selection is allowed in automatic mode.

After the menu is brought up and the variable MENU is set to allow selection, the next time MENUUP is called by a function button request, the short procedure which enters the user's menu selection is executed. Here, the menu is cleared from the display, and the Data Base Management module is called via a call to MDISP. MDISP will examine the COMMUN common as set by the MESLCT routine of the Menu module to determine what features need to be displayed and deleted.

MESLCT

MESLCT is the routine invoked by pushing the select function button; it is flowcharted in Figure 19. This routine records

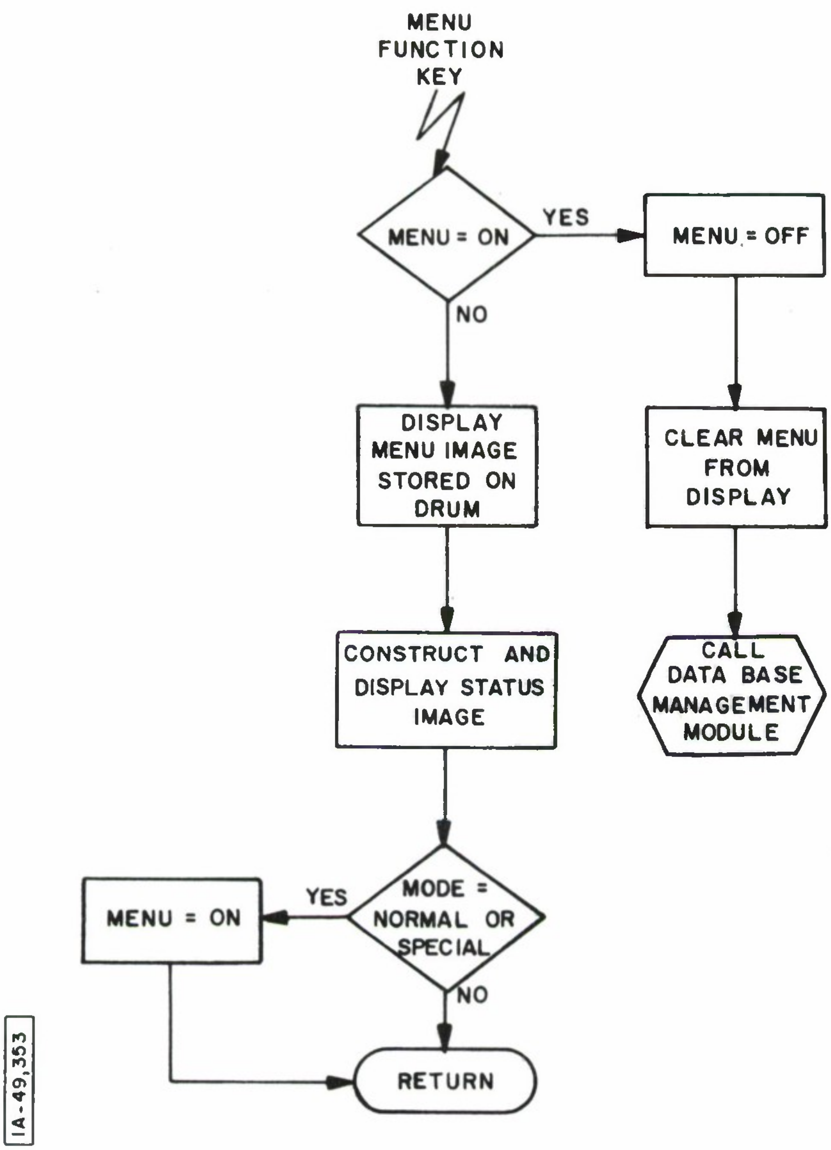


Figure 18 FLOWCHART OF MENUUP OF MENU MODULE

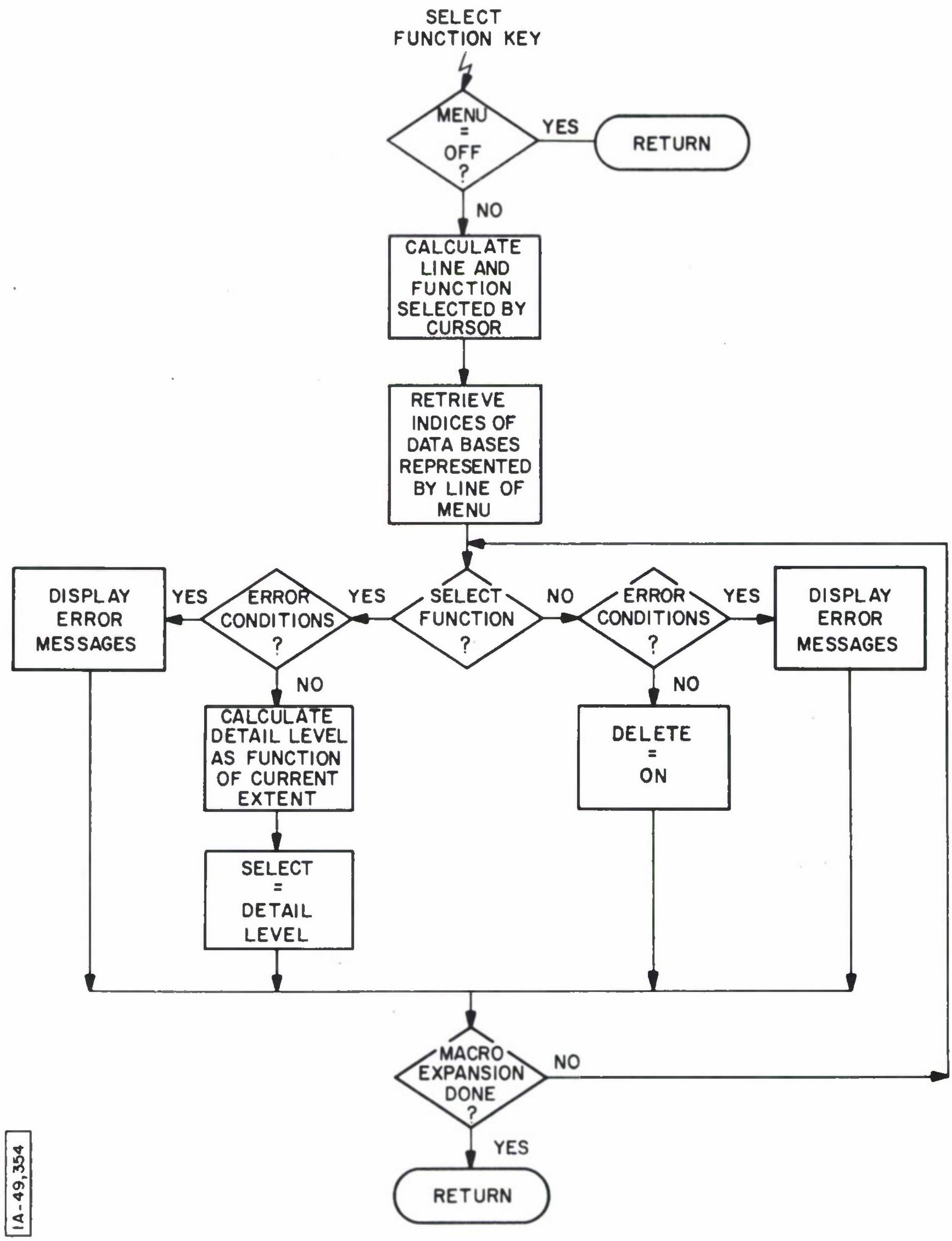


Figure 19 FLOWCHART OF MESLCT OF MENU MODULE

the features selected and deleted by the user in the COMMUN common and writes responses to the display screen each time the select function button is pressed. If the variable MENU is not set by MENUUP to allow user selection and deletion, MESLCT simply returns, resulting in a no-op.

On entry, MESLCT first determines which function the user has positioned the cursor beneath and also opposite which feature of the menu. An error message is displayed if the cursor is not aligned with either function or any of the features. If there is no error, the program is set up to process the macro expansion of the feature selected. This is done by setting up a loop which will be executed once for each feature in the expansion. The result is that each feature in the expansion appears as if it was selected separately.

If a feature was selected for display, several abnormal conditions are tested for - the feature has already been selected, it is currently displayed, or it has also been selected for deletion. In any of the three cases, a response is displayed. In the latter case, the entry in the DELETE array of the COMMUN common is turned off, resulting in a no-op for that feature. If none of these conditions exist, the subroutine CLEVEL determines at which detail level the feature should be displayed. The proper entry in the SELECT array of the COMMUN common is set equal to this detail level, a response is made to the user, and the detail level displayed in the menu.

The delete function works in an analogous fashion. A test is done to see if the feature is not currently displayed or if it has previously been selected for display. In either case an error message is displayed. In the last case SELECT is set to zero so that the feature will not be displayed. If no errors exist the DELETE entry for the feature being processed is set non-zero. An "X" opposite the feature is displayed in the menu and a response message given to the user.

FUNCTION REQUEST HANDLER

The Function Request Handler is invoked by pressing either the translate or zoom buttons. Its purpose is to perform an immediate zoom or translate on the data available, calculate a new center point and extent, and broadcast these parameters to the other processes in the system. To provide as fast a response to a user request as possible, the module is resident on the I-70, along with the PALLET routines that do translate and zoom.

Table I contains a list of the subroutines in the Function Request Module; a flowchart of the module is shown in Figure 20. ZOMTOP and TRANTP are the two mainline routines, the others are utilities used by one or both of the functions. The following discussion will be divided into two sections - one on zoom and one on translate.

Zoom

ZOMTOP is the mainline routine for both zoom in and zoom out requests. The direction of the zoom is determined by the magnitude of FAC, a calling parameter to ZOMTOP. When the zoom in button is pushed, the ZMINTP routine is invoked. This routine simply calls ZOMTOP with an appropriate value for FAC. For a zoom out, ZMOUTP is invoked, and ZOMTOP is called with a value for FAC that is the inverse of the value used for a zoom in. Thus, though ZOMTOP is the mainline routine, it is not directly invoked by the push of a function button.

Initially, ZOMTOP sets up the new center point and extent, message to be broadcast to the other processes in the system. This involves converting the absolute cursor position to map coordinates, calculating the new center point and extent, and storing these values in the CURSTA array.

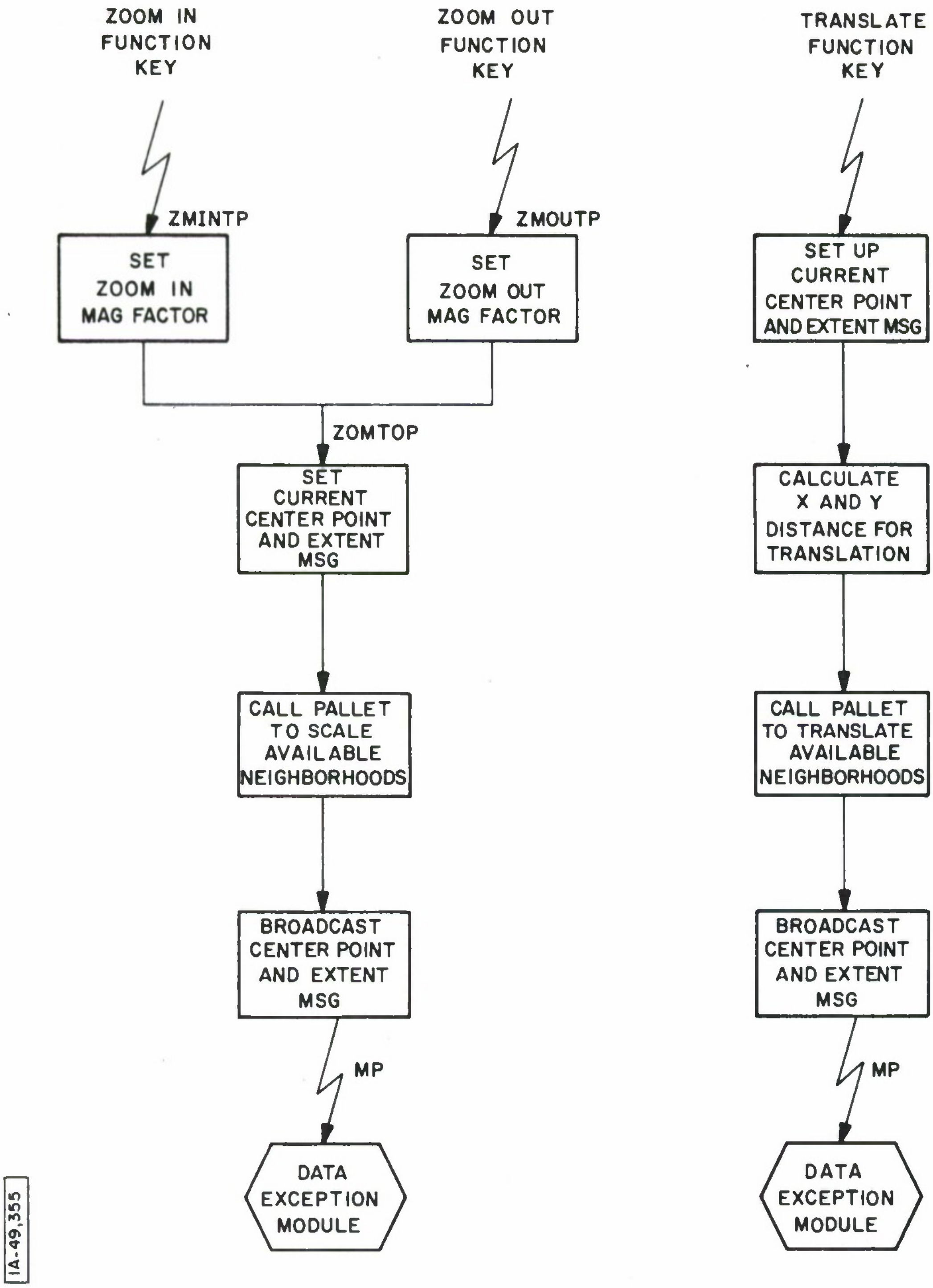


Figure 20 FLOWCHART OF FUNCTION REQUEST HANDLER

Once the message is set, the PALLET routine SCALE is called to perform an immediate zoom on the neighborhood of data available. The message is then sent via MP directly to the CURSTA routine of the Data Exception module and to any other process that wants to receive it.

Translate

TRANTP is invoked directly by a function key to handle a translate request. It, like ZOMTOP, sets the broadcast message first. It then calculates the distance the current display must be translated in the x and y directions. The PALLET routine TRANS is called to do this immediate translation. The new center point of the map is established as the cursor position, and the center point and extent message are broadcast. Again as in zoom, it is sent directly to the CURSTA routine of the Data Exception module.

DATA EXCEPTION MODULE

The Data Exception module determines which features currently displayed need a new neighborhood of data or a new detail level. It is entered only through the reception of a center point and extent message by the CURSTA routine, which in turn calls the mainline routine of the module, ZMTRNS. ZMTRNS examines all data bases for the possibility of a detail level change or a new neighborhood. The algorithms for detail level change are different for each of the three operating modes: automatic, normal and special. The following discussion begins with a brief explanation of the entry into the module and is followed by descriptions of the different zoom algorithms. It concludes with the translation algorithm. Table I contains a list of the subroutines included in this module; Figure 21 is a flowchart of the module.

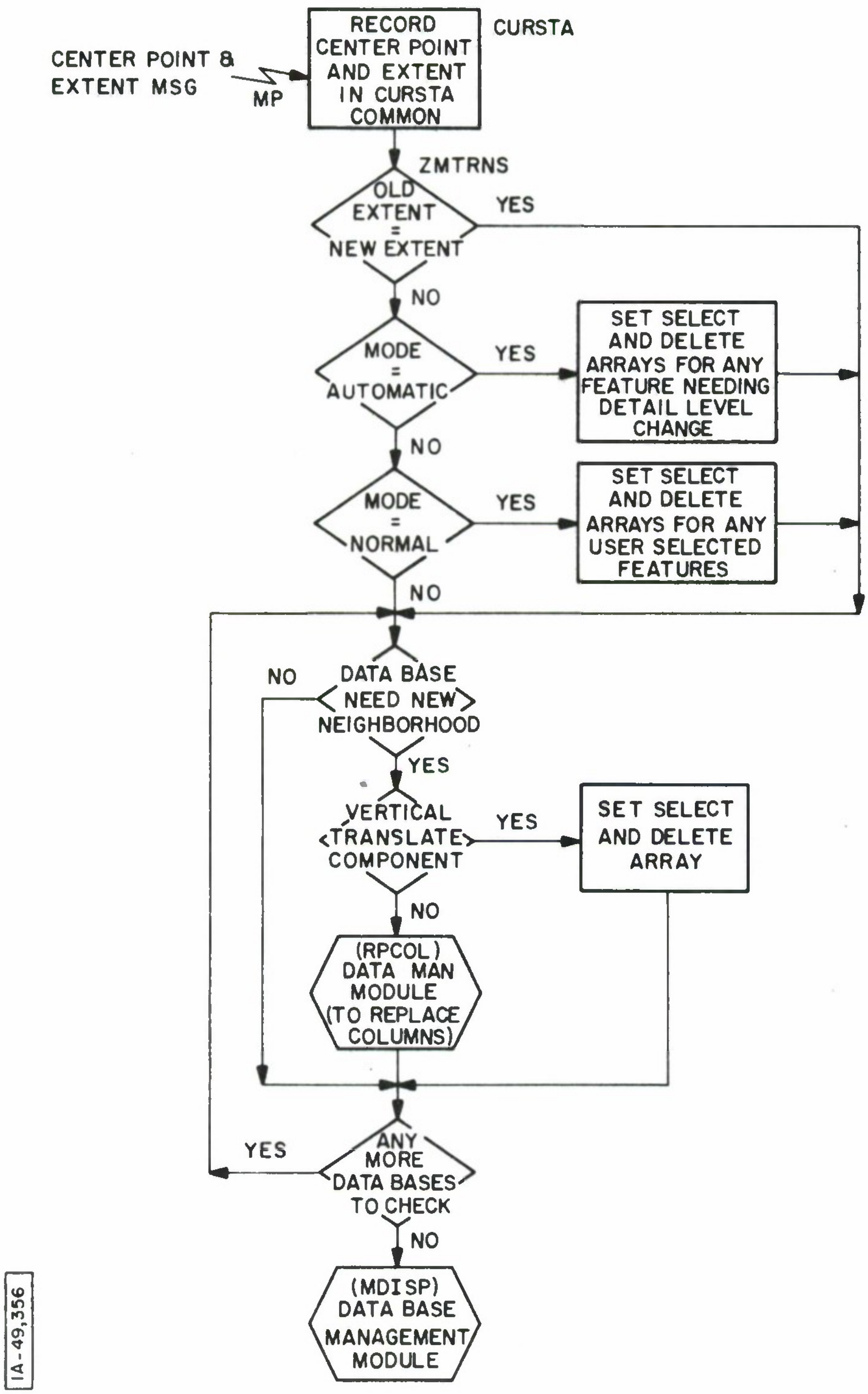


Figure 21 FLOWCHART OF DATA EXCEPTION MODULE

Module Entry

When a zoom or translate request is made the Function Request Handler broadcasts a message containing the new center point and extent and the previous center point and extent. The CURSTA routine of the Data Exception module receives this message and stores this global data in the CURSTA common for use by any routine. When the data is stored, ZMTRNS is called to determine new detail levels and neighborhoods.

Automatic Mode Zoom

If the system is in automatic mode, ZMTRNS calls AUTONZ to determine what features should be displayed and at what detail level. The AUTONZ algorithm is extremely simple. For each data base available to the system a call is made to the CLEVEL routine. This routine, using the predetermined thresholds for the detail levels of the data base being checked, calculates and returns the detail level at which the feature should be displayed. If the level returned is equal to the current level of the feature, nothing is done. Otherwise, the correct entry in SELECT is set equal to the returned level, and the appropriate entry in DELETE is set non-zero to force an erase of the current neighborhood of that feature. The procedure is repeated for each feature. Note that the question of whether a feature is displayed or not is strictly a function of the current extent and the predetermined thresholds of the detail levels of that feature.

Normal Mode Zoom

If the system is in normal mode, ZMTRNS will call the AUTOFZ routine. This routine functions in a manner analogous to AUTONZ with one addition. In AUTONZ the detail level of each data base is checked on every zoom. In AUTOFZ, only those data bases that have been selected are checked. A feature is considered selected if the location in the AUTOFZ array associated with it is non-zero.

This location is set when either the feature is selected or deleted by the operator using the menu, or when it comes within or falls out of its display range in automatic mode.

Special

In special mode no detail levels are changed. Thus, if the system is in special mode, ZMTRNS ignores detail level changes and calls the translation routine immediately.

Translation

After all potential detail level changes have been examined by ZMTRNS, MTRANS is called to determine if any neighborhoods need to be altered. Those features that had a detail level change require no checking, since their new neighborhoods will be calculated according to the new center point. In the case where the detail level of a feature was not changed, or where ZMTRNS was invoked by a pure translate request, neighborhoods must be checked for horizontal and vertical translation components. Horizontal translation requires the change of one or two columns; vertical translation requires an entirely new neighborhood.

For each data base, MTRANS uses the routine GRDCEN to determine the point of the grid used to divide a data base into blocks closest to the center point of the displayed map. This new grid point is compared to the grid point used to define the currently displayed neighborhood. If any change exists in the y coordinate, a vertical change has occurred, the entire neighborhood must be replaced and the proper entry in SELECT is set to the current detail level. The DELETE entry is also set non-zero.

If there is only a change in the x coordinate of the grid points, the RPCOL routine is called to invoke the Data Base Management module. This is a special entry into this module which only replaces one or two columns of a neighborhood at a time.

After all features have been checked, MTRANS calls the Data Base Management module via MDISP to replace all neighborhoods with a vertical translation component which have been tagged in the SELECT array of the COMMUN common by ZMTRNS.

DATA BASE MANAGEMENT

The Data Base Management module is responsible for retrieving data from secondary storage and displaying it, and erasing already displayed data from the display screen. It has two entry points - one for retrieving an entire neighborhood, and one for replacing only one or two columns of a neighborhood. These two entry points are MDISP and RPCOL, respectively. Both use the support routines listed in Table I and work in a very similar manner. The following discussion is in two parts - MDISP and RPCOL.

MDISP

MDISP is the Data Base Management module entry point which examines the COMMUN common, set by either the Menu module or the Data Exception module, to determine which data bases to delete from the display and which to retrieve and display. It first checks for any necessary erasures, and then proceeds to calculate, retrieve and display new neighborhoods, as diagrammed in Figure 22.

Each entry in the DELETE array of the COMMUN common is checked. If an entry is non-zero the CLMERS routine is called to make four entries in the erase table, one for each column of the neighborhood. After all data bases have been examined, the erase array is sent to PALLET on the display processor and the deletion process is completed.

Now each entry in the SELECT array of the COMMUN common is checked for a non-zero value. This non-zero value is the detail level at which the data base is to be displayed. For each

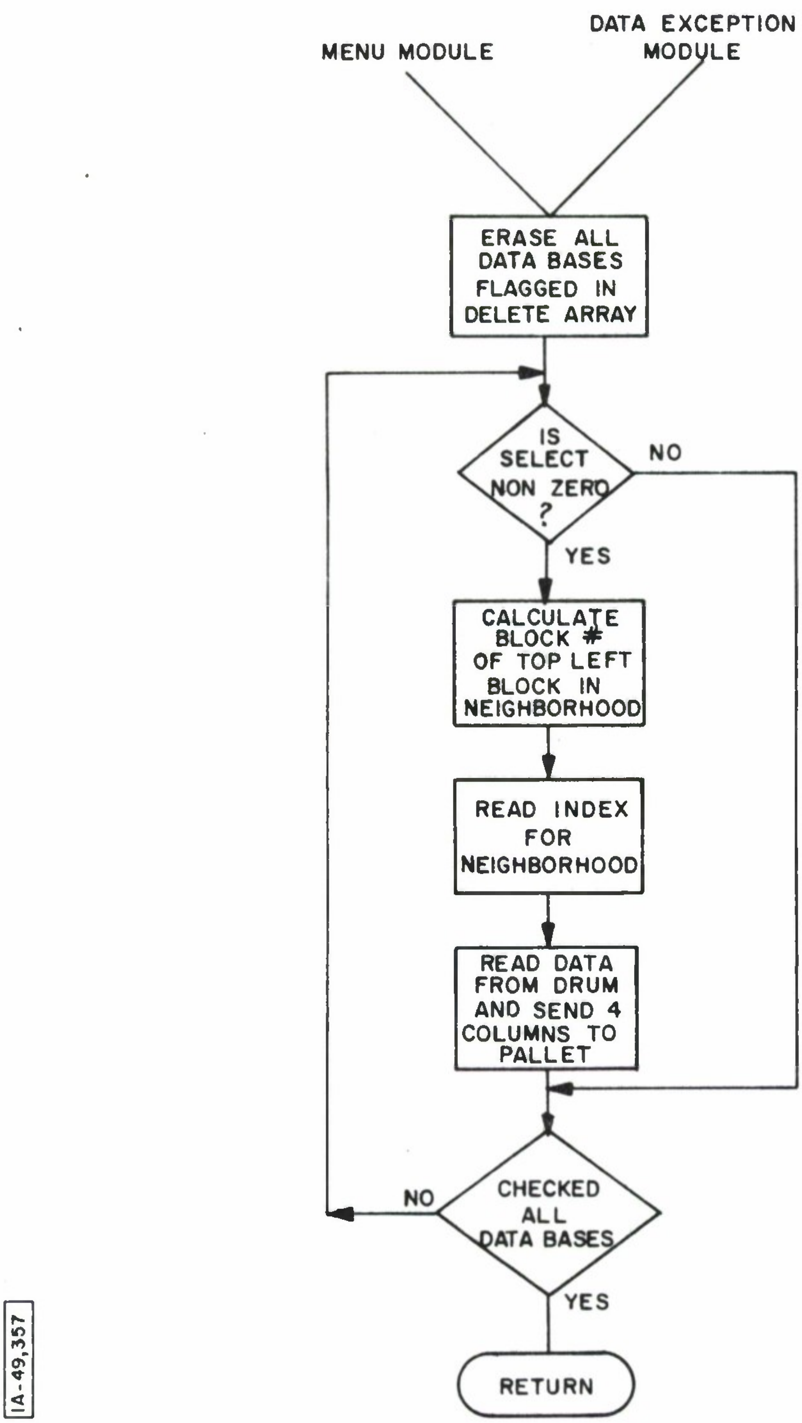


Figure 22 FLOWCHART OF MDISP ENTRY INTO DATA MANAGEMENT MODULE

feature selected, the center point of the neighborhood is calculated and used by the TOPLFT routine to determine the top left block of the 16 block neighborhood. This section is used by RINDEX to read the proper index entries from secondary store. Once the index is read, a core buffer is allocated large enough to hold the largest column of data as indicated by the index. The REDSND routine then uses each index entry to locate and read a column of data from secondary store. A PALLET image header is added to a retrieved column, and it is sent to PALLET to be displayed. This retrieval procedure is repeated once for each column in the data base. Once a neighborhood is displayed, the next feature selected is processed.

RPCOL

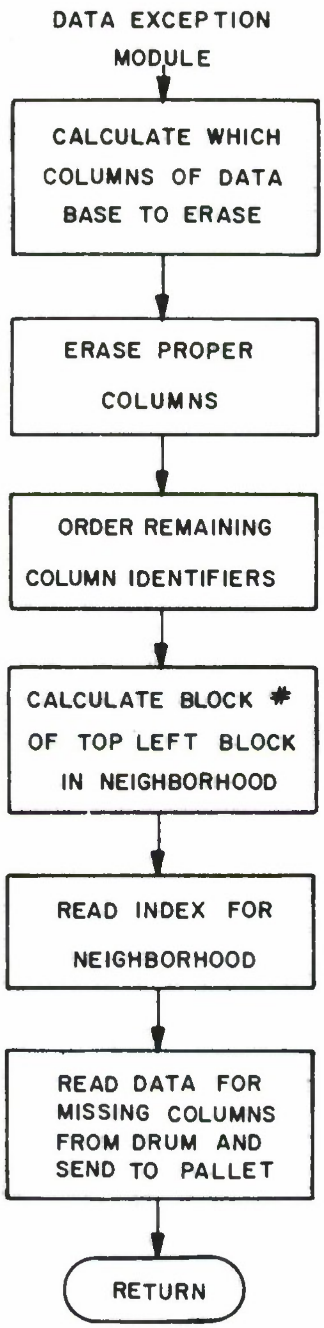
RPCOL is similar to MDISP in function except that it only works with one feature at a time. Like MDISP, it first erases already displayed columns, and then retrieves and displays the new column, or columns.

RPCOL first calculates which one or two columns are being replaced - the left-most, the right-most or the left two or the right two. The CLMERS routine is invoked to enter the proper columns in the erase array, and the array is sent to PALLET where the data is erased.

The index for the proper neighborhood is read. The retrieval process outlined in the MDISP section is now executed once for each column of the neighborhood not currently displayed. RPCOL is diagrammed in Figure 23.

MODE

The Mode module changes the mode of the system when either the automatic, normal or special function button is pressed. The module consists of only the three short routines listed in



IA-49,349

Figure 23 FLOWCHART OF RPCOL ENTRY INTO DATA MANAGEMENT MODULE

Table I. Each routine works exactly like the other two. When a mode function key is pressed, the respective routine is invoked which then sets the MODE variable of the CURSTA common to the proper value. The proper value is a 1, 2 or 3 depending on whether the mode selected is automatic, normal or special.

INITIALIZATION

The Initialization module prepares the system for execution. Its four routines are listed in Table I. Execution of the module is straightforward. Common variables are initialized by the REDCOM routines which read a tape made by the SETUP routine described in Appendix II. It initializes MP and FMP by setting up buffers and designating the file containing the menu image. It then assigns to the proper buttons the functions which are to be invoked when a function key is pressed. The initial center point and extent are sent via MP to the STATIN routine of the Function Request Handler to initialize the CURSTA common array on the I-70 side of the system. The Data Base Management module is finally called to display the initial data bases.

APPENDIX I

DATA BASE CONSTRUCTION PROGRAMS AND PROCEDURES

INTRODUCTION

In order to run the GDD, geographic data bases have to be stored on secondary store. This appendix gives a brief description of, and operating instructions for, the two programs needed to store prepared data bases on the Vermont drum.

DATA BASE CONSTRUCTION

Once a magnetic tape containing geographic data in chain form has been put through the detail analysis and editing process described in ESD-TR-76-360, "Geographic Data Base Development," it is ready to be stored on drum. (The format of one of these tapes and the programs used to manipulate the data are all fully described in ESD-TR-76-360.) This storage process for a single detail level of one feature is done in two steps by two programs — BLOKS and IMAGE. BLOKS divides a data base into the number of blocks specified by the user for that detail level, and IMAGE stores each block on drum and constructs an index.

BLOKS

The BLOKS program divides the chains of an edited tape of geographic data into the blocks which will be used to construct neighborhoods. Input to the program is (1) a geographic data base tape which has had each point of a chain assigned a detail rank, (2) the rank of points which the user wishes to extract from the tape for this detail level of a feature and (3) the number of blocks into which this detail level should be divided. The program then lays a grid over the map and examines one chain at a time. A chain that does not fall entirely within a block is broken up into smaller chains which do lie entirely in a single block. Only those points of a chain that have the same detail level or less than the one specified by the user are kept on the output tape. This output tape is a list of these new, smaller chains sorted by blocks. The blocks are ordered on the output tape according to columns, as dictated by the data base management scheme described in Section III. This process is repeated once for each detail level of a feature. By varying the detail level rank specified and the number of blocks, the number of points in the data base can be altered, and the geographic area covered by a single block can be changed.

BLOCKING ALGORITHM

The following algorithm is used by BLOKS to create the blocked output chains.

1. Input chains from the data base are processed one point at a time.
2. A grid block is assigned to the first point of a chain using the grid dimensions.
3. A chain is started in the assigned block, and the first point is filled into the chain.
4. Points are then read and copied into the grid block until either an end of chain mark is found (in which case the mark is written to end the chain in the current block and processing for the next chain is begun), or a point falls outside the current block.

5a. When a point falls outside the current block, the procedure described below is used to generate a block entry or exit point each time a grid line is crossed by the chain. (See Figure 24.)

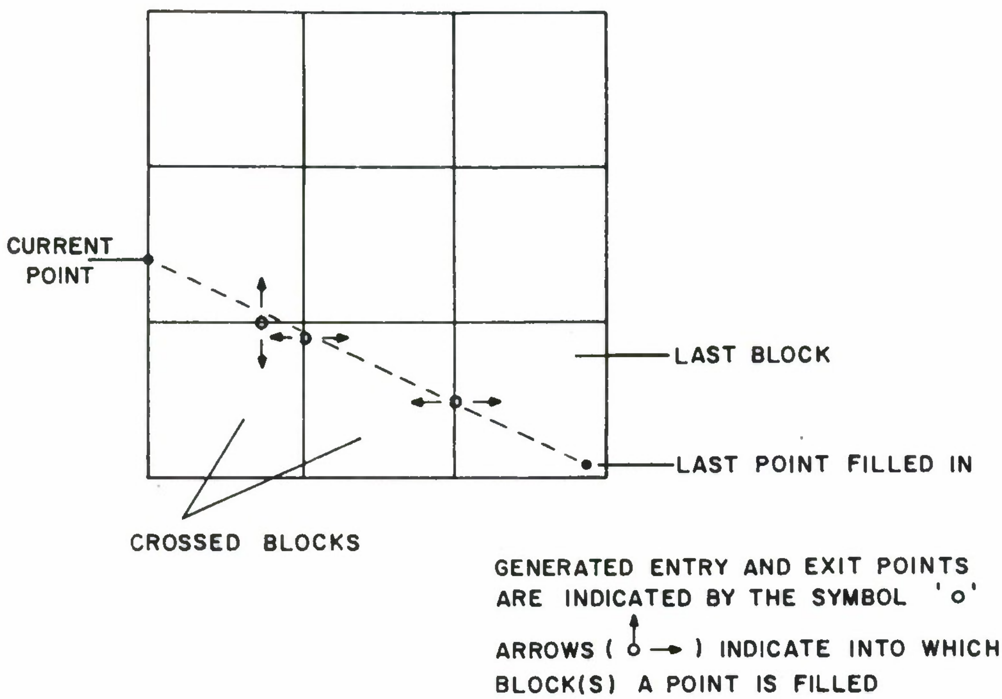
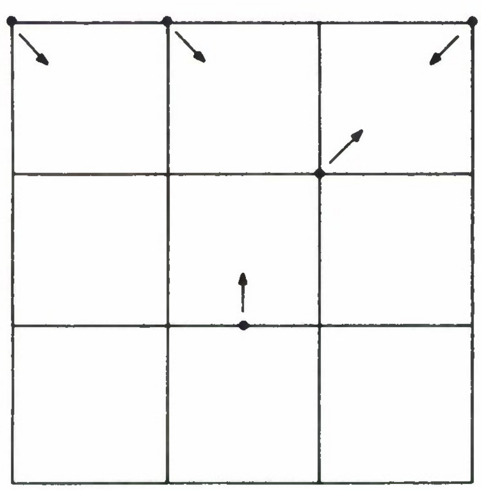


Figure 24 GENERATED ENTRY AND EXIT POINTS FOR CROSSED BLOCKS



IA - 49,433

Figure 25 ASSIGNMENT OF SINGLE - POINT CHAINS

b. The equation is found for the line connecting the current point (the point falling outside the current block) with the last point that was filled into a block.

c. The point at which this line crosses the boundary of the last block, the boundary crossing point, is filled into the block, and the chain is ended in this block.

d. A new chain is started in the block that has been entered, and the boundary crossing point becomes the first point in the new chain.

e. If the current point is in the crossed block it is now filled in as the second point in the new chain, and the next point is read as usual.

f. If the current point is outside the crossed block an exit point is generated for the block using the procedure in a and b above and the chain ended.

g. Entry and exit points are generated in this way for all blocks crossed in reaching the current point; i.e., until the current point falls into a crossed block.

A special procedure is used to handle input points which happen to fall on a boundary between grid blocks. Such a point will be referred to as a "boundary point."

6. When a boundary point falls outside the current block it is treated as a normal point for generating entry and exit points for crossed blocks. The boundary point is considered to be outside the block only if it is not on a boundary of <u>that</u> block.

7a. End of chain point -- If the boundary point is on a boundary of the current block and is the end of an input chain it is filled into the current block and the current chain is ended.

b. Mid-chain point -- If the boundary point is on a boundary of the current block and does not start or end a chain, the point is first filled into the current block. A look ahead to the next data point is then done to determine which block the input chain will enter.
If a new block is entered, the chain in the current block is ended and the boundary point is used to start a new chain in the block being entered. If the next point continues in the current block, the chain is processed normally.

c. Start of chain -- If the boundary point is on a boundary of the current block and is the start of an input chain, a look ahead to the next data point is done to determine which block the chain enters. A chain is started in this block using the boundary point as its first point.

d. Single point chain -- If a boundary point both starts and ends an input chain it is filled into the block assigned it by the block-assigning routine. This routine assigns a boundary point to the right or upper block depending on whether a vertical or horizontal grid line is straddled. A vertex point (one falling at the intersection of four blocks) is assigned to the upper right block. This rule is used unless it causes a point to fall outside the map box, in which case the point is assigned to the lower or left block. (See Figure 25.)

Operating Instructions

To run BLOKS two control cards must be supplied using the following formats:

Card #1 (corner points of map box)	lower left X coordinate	columns 1-8 (decimal in col. 3)
	lower left Y coordinate	columns 10-17 (decimal in col. 12)
	upper right X coordinate	columns 19-26 (decimal in col. 21)
	upper right Y coordinate	columns 28-35 (decimal in col. 30)
Card #2	number of blocks along the longer side of box	columns 1-3 (integer-right justified)
	detail level at which to select points	columns 5-6 (integer-right justified)

The load module for BLOKS is stored on tape MMC 001.

To run BLOKS this tape should be assigned a logical unit (6) and loaded using the operating system load command:

LO 6

The following units should be assigned to the appropriate devices:

LOGICAL UNIT	
01	card reader for control card input
02	output tape device
03	printer
04	drum file 4 - used for temporary storage of output data points
05	teletype
06	input tape device for map data base

BLOKS can now be started as follows:

ST 2E00

The control cards will be read first.

Then the teletype will ask:

ENTER 'T' OR 'B' FOR TOP OR BOTTOM OVERHANG

This means that the map area, defined by the four coordinates on the first input card, is not square. The longer side of the rectangle has been divided into the number of blocks requested on the second control card. The block overhang requested is the direction in which the shorter side of the map should be extended to allow an integer number of blocks with the same dimension as the blocks in the longer direction. The blocks are thus made square, having a side dimension equal to the length of long side divided by the number of blocks requested by user.

The grid dimensions and detail level to be used are now written to the printer. Then the data points are processed and totals for points and chains read in from the input tape and totals for points and chains actually selected are printed. Finally, the output tape is created and the table of blocks and the missing block messages (those blocks containing no data) are written on the printer.

IMAGE

From the output tape of BLOKS, the IMAGE program creates the drum file of blocks stored in column order and the row ordered index file to those blocks. Since each tape output by BLOKS contains data for only one detail level of a feature, IMAGE must be run once for each detail level of each feature. The method of operation is simple - IMAGE reads the input tape which BLOKS has created in column order and each block of data is stored on drum. As it is stored the address of the data and length of the data is recorded. Once all data is stored, the index is created by summing the lengths of the blocks in all possible groups of four which could form part of a neighborhood. These entries are then sorted in row order and stored in the index file.

Operating Instructions

IMAGE is the first program in drum file 51 and can be loaded with the system load sequence:

RW 51

LO 51

The following assignments are necessary:

LU	Device
01	Input tape
03	Printer
05	Teletype
07	Drum

Before executing, two drum files must be allocated to receive the data and the index. The program is executed with the system start command ST 2E00. IMAGE will respond with a set of questions, an example of which is given in Figure 26. The user responses are underlined. This data, input by the user, is formatted and printed on the line printer, followed by a list of the block numbers, in both row and column order, the drum address for the data of a block and the length in bytes of each block. This listing ends with the number of drum blocks used for the data file. A listing of the index is then printed. Figures 27, 28 and 29 show the three parts of the IMAGE output for a 16-block detail level of a geographic data base.

```
   MAPDATA
   NAME AND DETAIL LEVEL? (A4,1X,I2)
MAP  01
   DATA FILE NUM? (I3)
022
   INDEX FILE NUM? (I3)
024
   BLOCK COUNT? (I4)
0016
   X-AXIS BLOCK COUNT? (I2)
04
   END
   EOJ
```

Figure 26. Example Teletype Input for IMAGE Program

```
DATABASE MAP        DETAIL LEVEL  1
     DATA FILE IS  22
     INDEX FILE IS 24
     X-AXIS BLOCK COUNT  4
     Y-AXIS BLOCK COUNT  4
     TOTAL BLOCK COUNT  16
```

Figure 27. IMAGE Program Output Restating Input Parameters

ROW	COL	CHAINS	BLOCK	ENTRY	POINTS
13	1	3	Ø	Ø	33
9	2	5	2	1	2Ø
5	3	3	3	5	12
1	4	5	4	1	3Ø
14	5	2	5	15	1Ø
1Ø	6	9	6	9	71
6	7	6	11	Ø	47
2	8	8	13	15	41
15	9	7	16	8	52
11	1Ø	19	19	12	138
7	11	17	28	6	1Ø9
3	12	1Ø	35	3	41
16	13	14	37	12	91
12	14	1Ø	43	7	63
8	15	14	47	6	75
4	16	12	52	1	69

```
57 DRUM BLOCKS USED
 Ø OUT OF    16 BLOCKS MISSING
```

Figure 28. IMAGE Output Showing Number of Chains and Points per Map Block, Drum Block Address and Byte Entry in Drum Block for Each Block of Data.

ROW	BLOCK	ENTRY	POINTS	COL POINTS
13	0	0	33	95
14	5	15	10	169
15	16	8	52	340
16	37	12	91	298

1 DRUM BLOCKS USED BY INDEX

Figure 29. Output of IMAGE Program Showing Index for Column of Data

APPENDIX II

SYSTEM INITIALIZATION PROGRAM

INTRODUCTION

System initialization is done by the SETUP program. SETUP assigns an initial value to every variable in common and writes common out to tape. It also stores a PALLET image of a menu and an empty image of "world," to which the map will be attached, in the PALLET working file. When the GDD itself is executed, the tape created by SETUP is read into the common locations at the top of core, immediately initializing all variables in common.

In operation, SETUP reads the values for common variables from cards. Those common variables not required on an input card are set to zero or defined by some function of the input parameters. The following discussions will describe the input cards and the operating procedure for SETUP.

INPUT CARDS

A card (or several cards) is used to input the values for a common block. The order in which the commons are initialized is set by the program and will be specified below. The data on all input cards starts in column 10. From column 10 on, the format of the card varies according to its particular purpose. The first nine columns are not read by SETUP but can be used by the programmer to identify the card.

The first cards to be read contain values pertaining to the system as a whole or to all data bases. They include the FAC, MENU, TREES, COLORS, COMMUN, ERASE, MAP, and CURSTA commons. The next set of cards define the values of the DATBAS common. Finally, the PALLET file definition cards are read in and then the MACRO common is initialized.

Common Initialization Cards

The first ten cards are defined in Tables II through XI. In these tables, the first column gives the names of the variables to be initialized in the common identified at the top of the table. The information in the Purpose column can be supplemented from the common definition tables in Appendix IV. The example value is the value used by the current system.

Card #1 does not initialize a common. The length of the entire common section is used by SETUP to write the correct amount of core out to tape.

Card #3 initializes the variables which tell the system where to locate parts of the menu. The locations are given in a PALLET coordinate system defined to be(0.,0.) to (511.,479.) - one unit per dot on the screen. The locations are figured out by the 24 x 14 dot matrices which contain a character. Thus, the first line on the bottom of the screen has a y-coordinate of 24 and the first character has an x- coordinate of 0. The second line has y- coordinate = 48 and the second character has an x- coordinate of 14.

On card #5 the function buttons are identified. The initial values given in the table are the decimal representations of the characters generated by the RAMTEK when these function buttons are hit. These characters are defined in the RAMTEK documentation.

Card #6 defines the colors, red, yellow, green and black, used by the menu for the one plugging of the RAMTEK given in Appendix III. These colors can be changed by replugging the red, blue and green outputs of the RAMTEK into different plugs on the TV monitor. This, too, is documented in the RAMTEK manuals.

The COMMUN common is initialized by card #7. The feature data bases are numbered in the order in which they are read into the DATBAS common. By setting the proper entries in the SELECT array to the detail level of the features needed in the initial display, the user forces the display of these features.

DATBAS Initialization

The next group of cards to be read in defines the feature library by initializing the DATBAS common. This process is done in a loop repeated once for each feature. Within this loop is another loop repeated once for each detail level of the feature. Thus, each feature is defined by one card (see Table XII) followed by two cards (see Tables XIV and XV) for each detail level of that feature. This sequence is repeated for each feature. The order in which the features are read in is the order in which the features are indexed throughtout the GDD program. If the first feature read in is coastlines, then to select coastlines for display, the first element of the SELECT array in the GOMMUN common is set to the desired detail level.

Tables XII through XV define the cards needed to perform the DATBAS initialization. The first card in Table XII is needed only once to define the number of features in the library. Each of the other three types of cards must be repeated to initialize all features. If there are two features, the first with two detail levels and the second with one, the order of cards is as follows:

Card #11
Card #12 for 1st feature
Card #13 for 1st detail level
Card #14 for 1st detail level
Card #13 for 2nd detail level
Card #14 for 2nd detail level
Card #12 for 2nd feature
Card #13 for 1st detail level
Card #14 for 1st detail level

PALLET File and Macro Definition

This last group of cards initializes the PALLET file containing the menu and the macro expansion capability of the GDD. Two cards (Table XVI and XVII) are read first, followed by a loop which reads two cards (Tables XVIII and XIX) for each entry in the menu, in the order in which they should appear in the menu.

OPERATING INSTRUCTIONS

The program SETUP can be loaded from tape DHL 007 using the COREDP program. The following sequence is necessary to load SETUP. Computer responses are underlined.

Load DHL 007 on drive 95.

AS	0195
RW	DE
BI	DC00
LO	DE
ST	DC00

LOAD OR STORE

LO

DEVICE NUMBER (NN)

01

START, END

0080, 7000

EOJ

The following assignments must be made to run SETUP:

LU	Device
01	Card Reader
05	Teletype
06	Output tape
07	Drum

After the assignments are made, load the input cards into the card reader and issue the start command:

ST 2E00

When SETUP is done, the output tape will contain the initialized common. The tape may now be used to initialize the GDD.

Table II

Card #1

Common - NA

Format Statement (9X,I4)

Variable	Purpose	Example Initial Value	Card Col	Format
ICOML	length in decimal of all common	2844	10-13	I4

Table III

Card #2

Common - FAC

Format Statement (9X,2F10.0)

Variable	Purpose	Example Initial Value	Card Col	Format
ZOOMIN	magnification factor when zooming in	.666666	10-19	F10.0
ZOOMOT	magnification factor when zooming out	1.5	20-29	F10.0

Table IV

Card #3

Common - MENCON

Format Statement (9X,7F6.1)

Variable	Purpose	Example Initial Value	Card Col	Format
ONXC	x-coordinate of the left side of the ON column of the menu	28.0	10-15	F6.1
ONRXC	x-coordinate of the right side of the ON column	304.	16-21	F6.1
OFFXC	x-coordinate of left side of OFF column	350.	22-27	F6.1
OFFRXC	x-coordinate of right side of OFF column	392.	28-33	F6.1
STATY	y-coordinate of status line of menu	48.	34-39	F6.1
RESYC	y-coordinate of system response line	24.	40-45	F6.1
RLEFT	x-coordinate for start of status and response lines	28.	46-51	F6.1

Table V

Card #4

Common - MENCON

Format Statement (9X,4(2A4,1X))

Variable	Purpose	Example Initial Value	Card Col	Format
MENNME	contains PALLET name of the image of the menu	MENUIMGE	10-17	2A4
STATUS	contains PALLET name of the image containing status information	STATUS	19-26	2A4
SYSTAT	contains PALLET name of character string of the system status message	SYSTATUS	28-35	2A4
SYSRES	contains PALLET name of character string of the response message	SYSRES	37-44	2A4

Table VI

Card #5

Common - TREES

Format Statement (9X,2I1,1X,2A4,8(I3,1X),2A4)

Variable	Purpose	Example Initial Value	Card Col	Format
MAPTRE	PALLET device number of map display tree	1	10	I1
MENTRE	PALLET device number of menu display tree	2	11	I1
WORLD	PALLET name of node to which geography is attached	WORLD	13-20	2A4
ZINBUT	Zoom in function button on RAMTEK	141	21-23	I3
TRNBUT	Translate function button	134	25-27	I3
SLCTBT	Select function button	139	29-31	I3
AUFBUT	Normal mode selection button	140	33-35	I3
ZOTOUT	Zoom out function button	137	37-39	I3
MENBUT	Menu function button	135	41-43	I3
AONBUT	Automatic mode selection button	136	45-47	I3
STABUT	Special mode selection button	144	49-51	I3
RMAP	Contains PALLET name for image, defined in map coordinate system, containing geographic data	MAP	53-60	2A4

Table VII

Card #6

Common - COLORS

Format Statement (4(I2,1X))

Variable	Purpose	Example Initial Value	Card Col	Format
RED	decimal representation for color red on RAMTEK	02	10-11	I2
YELLOW	RAMTEK color yellow	06	13-14	I2
GREEN	RAMTEK color green	04	16-17	I2
BLACK	RAMTEK null color	00	19-20	I2

Table VIII

Card #7

Common - COMMUN

Format Statement (9X,10I2)

Variable	Purpose	Example Initial Value	Card Col	Format
SELECT	defines which feature data bases should initially be displayed by setting the proper entries in the SELECT array to the wanted detail level.	01	10-29	10I2

Table IX

Card #8

Common - ERASE

Format Statement (9X,I3)

Variable	Purpose	Initial Value	Card Col	Format
ERSIZE	defines the length of the ERASE array	40	10-12	I3

Table X

Card #9

Common - MAP

Format Statement (9X,4F10.4)

Variable	Purpose	Example Initial Value	Card Col	Format
MX1	x-coordinate of lower left corner of European map in projected map coordinates	-.398	10-19	F10.4
MY1	y-coordinate of lower left corner of European map in projected map coordinates	-.266	20-29	F10.4
MX2	x-coordinate of upper right corner	.278	30-39	F10.4
MY2	y-coordinate of upper right corner	.410	40-49	F10.4

Table XI

Card #10

Common - CURSTA

Format Statement (9X,4F10.4)

Variable	Purpose	Example Initial Value	Card Col	Format*
XCENM	x-coordinate of the center of the map for initial display in projected map coordinates	-.03	10-19	F10.4
YCENM	y-coordinate of initial center	.03	20-29	F10.4
XEXTNT	initial x-extent of map in map units per dot on display screen	.002093749	30-39	F10.4
YEXTNT	initial y-scale of map in map units per raster line	.001601576	40-49	F10.4

*The FORTRAN input routine gives precedence to the decimal point in the input field, overriding the format specified for that field.

Table XII

Card #11

Common - DATBAS

Format Statement (9X,I2)

Variable	Purpose	Example Initial Value	Card Col	Format
NUMDB	defines number of features in library	2	10-11	I2

Table XIII

Card #12 (and repeated for each feature)

Common - DATBAS - one card for each feature

Format Statement (9X,A4,1X,I1,1X,I1,1X,I2)

Variable	Purpose	Example Initial Value	Card Col	Format
PREFIX	contains data base name	RIVR	10-13	A4
NUMLEV	number of detail levels for this feature	2	15	I1
INMENU	0 if feature not listed in menu 1 if feature is listed	1	17	I1
POSFET	position feature is listed in menu in lines from the top of the list	2	19-20	I2

Table XIV

Card #13 (and repeated for each detail level)

Common - DATBAS (one card for each detail level of feature currently being initialized)

Format Statement (9X,2F10.6,2I4)

Variable	Purpose	Example Initial Value	Card Col	Format*
ZMOTHR	zoom out extent threshold	.001395832	10-19	F10.6
ZMINTH	zoom in extent threshold	.000275720	20-29	F10.6
NUMX	number of blocks into which detail level is divided in x direction	9	30-33	I4
NUMY	number of blocks y axis is divided into y direction	9	34-37	I4

---

*ibid.

Table XV

Card #14 (and repeated for each detail level)

Common - DATBAS (one card for each detail level of feature currently being initialized)

Format Statement (9X,2I3,I1,1X,I2)

Variable	Purpose	Example Initial Value	Card Col	Format
IFILE	decimal file number of drum file containing data	119	10-12	I3
DBINDX	decimal file number of drum file containing index	120	13-15	I3
ITYPE	1=point data base 2=line data base	2	16	I1
ICOLOR	color of data base according to current plugging of RAMTEK	08	18-19	I2

Table XVI

Card #15

Common - FILE

Format Statement (9X,I3)

Variable	Purpose	Example Initial Value	Card Col	Format
MFILE	identifies decimal drum file number to be used by PALLET to store menu image	115	10-12	I3

Table XVII

Card #16

Common - MACRO

Format Statement (9X,I2)

Variable	Purpose	Example Initial Value	Card Col	Format
NUMFET	number of features actually listed in the menu	2	10-11	I2

Table XVIII

Card #17 (and repeated for each line in the menu)

Common - MACRO

Format Statement (9X,4A4,I2)

Variable	Purpose	Example Initial Value	Card Col	Format
TITLE	16 characters to appear as the menu entry for that feature	RIVERS(2)	10-25	4A4
MACNUM	number of features in the macro expansion of this menu entry	1	26-27	I2

Table XIX

Card #18 (and repeated for each line in the menu)

Common - MACRO

Format Statement (9X,4I2)

Variable	Purpose	Example Initial Value	Card Col	Format
MACEXP	the ordered numbers of each feature data base represented by this line in the menu	01	10-17	4I2

APPENDIX III

GDD OPERATING PROCEDURES

OPERATING INSTRUCTIONS

The following three tapes are needed to run the GDD:

DHL 019 - contains core image of GDD for the I-70

DHL 004 - contains core image of GDD for the I-4

DHL 018 - output of SETUP program to initialize the I-4

Initialize both machines and start at address X'108'. Be sure the RAMTEK is on and plugged in the following manner:

CHANNEL	SUBCHANNEL	DISPLAY MONITOR INPUT	MENU MONITOR INPUT
0	1	R	
0	2	G	
0	3	B	
1	1,0		R
1	2,0		G
1	3,0		B

The system tapes can now be loaded using the COREDP program. The following sequences are necessary: (computer responses are underlined)

I-70		I-4	
Load DHL 019 on drive 85		Load DHL 004 on drive 95	
AS	0685	AS	0195
ST	2E00	RW	DE
	LOAD OR STORE	BI	DC00
LO		LO	DE
	DEVICE NUMBER	ST	DC00
06			

I-70 (cont'd)

START,END

0080, 8000

EOJ

I-4 (cont'd)

LOAD OR STORE

LO

DEVICE NUMBER

01

START,END

0080, FFFE

EOJ

Now load DHL 018, the SETUP output tape, on drive 85 on the I-4. Start the I-70 with the following command:

ST 3000

Then start the I-4 by issuing :

ST 2E00

Both machines should type MPV2.3 followed by the word SYSINIT on the I-4. The map should then appear on the display screen. When the entire map is displayed, both machines will cycle with the display panel lights blinking, indicating they are idling waiting for messages.

Once the display has appeared, the operator can zoom, translate, use the menu, or select modes as described in Section II.

APPENDIX IV

COMMONS

INTRODUCTION

The GDD has 13 labeled common blocks. The variables are grouped in blocks according to function - variables relating to a specific aspect of the system are in one common block. Nine of the thirteen blocks contain only static variables which retain their initial values throughout the operation of the GDD. The other four either contain system status information or are used for intermodule communication. The discussion below will center on these dynamic commons; the information in the static commons is briefly stated at the end.

COMMONS

The use of the COMMUN common to provide intermodule communication has been discussed in Section V. The use of SELECT and DELETE arrays of the COMMUN common is restated in Table XX. What has not been stated before is the relationship between a feature data base and an entry in the SELECT and DELETE arrays or any of the arrays in the DATSTA or DATBAS commons. Previously, only the "proper entry" has been referred to. The answer is simple and relates to any variable array containing information about the set of feature data bases: when the system is initialized data is read from cards describing the size and location of each feature data base. The order in which the description of the data base is read is the order in which it appears in the common arrays. The first feature read in becomes feature number one, and the first entry in all arrays pertaining to the feature data bases is assigned to feature number one. For example, in the case of the COMMUN common, if the river data base is to be displayed at level one and is currently displayed at level two, DELETE(2) is set non-zero and SELECT(2) is set to one. Assuming the river feature was the second feature described by the cards at initialization time, the current river neighborhood will be erased and a new one retrieved from detail level 1.

CURSTA is another dynamic common briefly discussed in Section V. It contains the current and previous status of the display window - center point, extent, cursor position and mode. Table XXI lists the CURSTA variables and their meaning. All variables of the CURSTA common except MODE are only altered by the CURSTA subroutine of the Data Exception module when it receives a message from the Function Request module. This message contains not only the new center point and scale, but also previous values. All values are stored in the CURSTA common. The variable MODE indicates whether the system is in automatic, normal or special mode and is only changed by the three routines in the Mode module.

DATSTA is another status common. It contains the status of each data base currently displayed. Table XXII lists the variables and the meaning of the DATSTA common. CURLEV and COL are modified only by the Data Base Management module, GX and GY by the Data Exception and the Data Base Management module and AUTOFS by the Menu module and Data Exception module.

The final dynamic common is ERASE. This common contains all variables needed to erase a column or set of columns from the display screen. Table XXIII defines the variables in this common.

STATIC COMMONS

The static commons contain constants defined when the system is initialized. No variable in the common is altered after initialization. The purpose of each of the nine static common blocks is self-evident from Table XXIV through Table XXXII which define the variables in each common. Only the MACRO common needs elucidation.

The MACRO common contains all the variables necessary for identifying which feature has been selected from the menu by the user and expanding this feature into as many as four different data bases. For example, the menu could contain separate entries for coastline and political boundaries; each could be turned on or off separately; or, either in addition to or in place of those two entries, an entry called "boundaries" could appear in the menu. If "boundaries" were selected it would be expanded into the two data bases, coastlines and political boundaries. This expansion would be done by first examining MACNUM to determine how many data bases are represented by the feature selected from the menu. In this case, it is two. The first two entries in MACEXP for the menu feature selected are the ordered numbers assigned at initialization time to political and coastline boundaries. These numbers are used as the indices of the SELECT and DELETE arrays of the COMMUN common to request an operation on these data bases. It should be noted that the index into MACNUM and MACEXP is the position of the selected feature on the display screen.

Table XX

COMMUN COMMON

VARIABLE	TYPE	MEANING
SELECT(10)	I	Intermodule communication identifying which data bases have been selected for display either by the menu or automatic zoom thresholds. 0 - data base is not to be displayed 1 - 4 detail level at which data base should be displayed.
DELETE(10)	I	Intermodule communication identifying which currently displayed data bases should be deleted either because of menu deletion or automatic zoom thresholds. 0 - if not to be deleted 1 - if to be deleted.

Table XXI

CURSTA COMMON

VARIABLE	TYPE	MEANING
MODE	I	Identifies the mode of the system 1 = Automatic 2 = Normal 3 = Special
XCENM	R	X-coordinate of map center in projected map coordinates
YCENM	R	Y-coordinate of map center in projected map coordinates
OXCEN	R	Value of XCENM prior to last translate or zoom
OYCEN	R	Value of YCENM prior to last translate or zoom
XEXTNT	R	Current extent of displayed map in X direction given in map units per dot on screen
YEXTNT	R	Current extent of displayed map in Y direction given in map units per lines on the screen
OXXTNT	R	Value of XEXTNT prior to last zoom or translate
OYXTNT	R	Value of YEXTNT prior to last zoom or translate
XCURA	R	X position of cursor in absolute device coordinates (0-479)
YCURA	R	Y position of cursor in absolute device coordinates (0-511)
OXCURA	R	Value of XCURA prior to last zoom or translate

Table XXI (concluded)

VARIABLE	TYPE	MEANING
OYCURA	R	Value of YCURA prior to last zoom or translate
XCURM	R	X cursor position in map coordinate system
YCURM	R	Y cursor position in map coordinate system
OXCURM	R	Value of XCURM prior to last zoom or translate
OYCURM	R	Value of YCURM prior to last zoom or translate

Table XXII

DATSTA COMMON

VARIABLE	TYPE	MEANING
CURLEV(10)	I	The current detail level at which each feature data base is displayed. 0 - not displayed 1 - 4 current displayed detail level
AUTOFS(10)	I	0 - if feature not currently selected for display by menu or automatic mode. 1 - if feature currently selected for display by menu or automatic mode. If AUTOFS = 1 for a feature the feature is not necessarily displayed; this is still a function of the extent thresholds for that feature. It does mean that if in normal mode and AUTOFS = 1, the feature will be displayed when scale is within the thresholds.
GX(10)	I	X-coordinate of grid point which defines center of the current neighborhood of blocks. If a detail level is divided into N blocks in the X direction, GX for a data base ranges from 2 to N-2 depending on which line of the grid the center point of the displayed map is nearest.
GY(10)	I	Y-coordinate of grid point which defines center of the current neighborhood of blocks. If a detail level is divided into N blocks in the Y direction, GY for that data base ranges from 2 to N-2 depending on which vertex of the grid the center point of the displayed map is nearest.
COL(4,10)	I	For each displayed feature, COL contains the block numbers of the four blocks of the neighborhood which are at the top of the columns of the neighborhood. The blocks are numbered in row order.

Table XXII (concluded)

VARIABLE	TYPE	MEANING
COL (cont'd)		COL holds these block numbers in their order in the neighborhood from left to right. A value of 0 indicates that a column in that position contains no data. If a feature is not displayed, COL for that feature is 0.

Table XXIII

ERASE COMMON

VARIABLE	TYPE	MEANING
ERSAR(2,40)	A	Contains a list of the 8-character PALLET names of the columns of neighborhoods that need to be erased from the display. A name consists of the 4 character feature name in the variable PREFIX and the four byte column number in the variable COL.
ICNT	I	Counts the number of entries currently in ERSAR.
ERSIZE	I	Maximum size of the ERSAR array.

Table XXIV

COLORS COMMON

VARIABLE	TYPE	MEANING
RED	I	Value needed to produce red on the RAMTEK for standard plugging given in Appendix III.
YELLOW	I	Same as above for yellow
GREEN	I	Same as above for green
BLACK	I	Value is 0 to produce black.

Table XXV

DATBAS COMMON

VARIABLE	TYPE	MEANING
PREFIX(10)	A	Contains the 4-character feature name to be used in constructing PALLET names of displayed images.
NUMLEV(10)	I	Number of detail levels in each feature.
INMENU(10)	I	1 - feature is listed in the menu 0 - feature is not listed in the menu but is included in the macro expansion of some other listing in the menu.
ZMOTHR(4,10)	R	The X extent values at which, when zooming out, the detail level of a feature should be changed.
ZMINTH(4,10)	R	The X extent values at which, when zooming in, the detail levels of a feature should be displayed or changed.
D(4,10)	R	The width in map units in the X direction of a single block of each of the possible detail levels of a feature.
NUMX(4,10)	I	Number of blocks in X direction into which each of the four possible detail levels of a feature is divided.
NUMY(4,10)	I	Number of blocks in Y direction into which each of the four possible detail levels of a feature is divided.
IFILE(4,10)	I	Decimal drum file number for the data base file of each of the four possible detail levels of a feature.
DBINDX(4,10)	I	Drum file number for each of the index files of the four possible detail levels of a feature.

Table XXV (concluded)

VARIABLE	TYPE	MEANING
ITYPE(4,10)	I	Data base type as defined by PALLET 1 = point data base 2 = line data base
ICOLOR(4,10)	I	Color with which each of the four possible detail levels of a feature should be displayed. Value is determined by the RAMTEK plugging as explained in Appendix III.
NUMB	I	Actual number of feature data bases available to the system up to a maximum of 10.

Table XXVI

FAC COMMON

VARIABLE	TYPE	MEANING
ZOOMIN	R	Factor with which the old extent must be multiplied to give new extent after a zoom in.
ZOOMOT	R	Factor with which the old extent must be multiplied to give new extent after a zoom out.

Table XXVII

FILE COMMON

VARIABLE	TYPE	MEANING
MFILE	I	Decimal drum file number of file to be used by PALLET for storage of image definitions.

Table XXVIII

MACRO COMMON

VARIABLE	TYPE	MEANING
MACNUM(10)	I	Number of data bases in the macro expansion of each line of the menu list. There are a total of 10 possible lines in the menu. There is a maximum of 4 features in a macro expansion.
MACEXP(10,4)	I	For each of the lines in the menu list, MACEXP contains the index of the feature data bases represented by that line. The actual number of features for each line is determined by MACNUM. The index for a feature in the macro expansion is the order in which the the data bases are defined during initialization.
FETPOS(10)	I	For each line on the screen, the value of FETPOS gives the proper index into MACNUM and MACEXP. (The line numbers on the screen do not directly give the index since there are several title lines in the menu.)
POSFET(10)	I	For a given feature, POSFET contains the line of the menu on the screen which represents that feature. It is in a sense the reverse of FETPOS. POSFET goes from feature to screen, FETPOS goes from screen to macro index.
NUMFET	I	Number of lines in the menu list of features.

Table XXIX

MAP COMMON

VARIABLE	TYPE	MEANING
MX1	R	X coordinate of the lower left corner of the map area in projected map coordinates.
MY1	R	Y coordinate of the lower left corner of the map area in projected map coordinates.
MX2	R	X coordinate of the upper right corner of the map area in projected map coordinates.
MY2	R	Y coordinate of the upper right corner of the map area in projected map coordinates.

Table XXX

MENCON COMMON

VARIABLE	TYPE	MEANING
MENU	I	0 - if menu is not currently displayed 1 - if menu is currently displayed
ONXC	R	X coordinate of left margin of ON column of the menu*
ONRXC	R	X coordinate of the right margin of ON column of the menu*
OFFXC	R	X coordinate of left margin of OFF column in menu*
OFFRXC	R	X coordinate of the right margin of OFF column in menu*
MENNME(2)	A	8-character PALLET name of menu image
STATUS(2)	A	8-character PALLET name of status image
SYSTAT(2)	A	8-character PALLET name of status character string
SYSRES(2)	A	8-character PALLET name of system response character string
STATY	R	Y coordinate of location of status message in menu*
RESYC	R	Y coordinate of location of response message in menu*
RLEFT	R	X coordinate of start of both status and response message*

*Defined in terms of menu coordinate system - lower left(0,0) and upper right (511,479).

Table XXXI

MNUTIA COMMON

(constant variables used only to make code more readable)

VARIABLE	TYPE	MEANING
ON	I	1
OFF	I	0
YES	I	1
NO	I	0
UP	I	1
PASS	I	-1
AUTON	I	1
AUTOF	I	2
STATIC	I	3

Table XXXII

TREES COMMON

VARIABLE	TYPE	MEANING
MAPTRE	I	PALLET device number on which map is displayed
MENTRE	I	PALLET device number on which menu is displayed
WORLD(2)	A	8-character PALLET name of tree node to which map is attached
ZINBUT	I	RAMTEK function key for zooming in
TRNBUT	I	RAMTEK function key for translating
SLCTBT	I	RAMTEK function key for menu selection
AUFBUT	I	RAMTEK function key for selecting normal mode
ZOTBUT	I	RAMTEK function key for zooming out
MENBUT	I	RAMTEK function key for requesting and entering menu
AONBUT	I	RAMTEK function key for selecting automatic mode
STABUT	I	RAMTEK function key for selecting static mode
RMAP(2)	A	8-character PALLET name of image containing map data

APPENDIX V

PROGRAM SUMMARY SHEETS

G D D

PROGRAM SUMMARY SHEET

1) ROUTINE: ABIND 2) MODULE: INITIALIZATION 3) MACHINE: I-4

4) CALLING STATEMENT:

NA

5) ARGUMENTS:

NA

6) CALLED BY:

NA

7) CALLS ROUTINES:

NA

8) COMMONS REFERENCED:

NA

9) PURPOSE AND METHOD:

ABIND is a dummy routine used during linking to account for entry points called by Pallet but not needed by the GDD. By not including these Pallet routines core was saved.

G D D

PROGRAM SUMMARY SHEET

1) ROUTINE: ALLOC
DEALOC

2) MODULE: DATA BASE MANAGEMENT

3) MACHINE: I-4

4) CALLING STATEMENT:

CALL ALLOC(INDEX,IPNT,NONE)

5) ARGUMENTS: INDEX - the unpacked index for the neighborhood to be retrieved IPNT - returned pointer to allocated core block ; NONE - returned flag indicating empty neighborhood

6) CALLED BY:

SETINX

7) CALLS ROUTINES:

NA

8) COMMONS REFERENCED:

NA

9) PURPOSE AND METHOD:

ALLOC allocates a core buffer large enough to hold the longest column of the neighborhood being retrieved. DEALOC deallocates the currently allocated buffer. ALLOC compares the lengths of the four columns of the neighborhood to determine which is longer. Since the length is the number of points in the column, the buffer must be 8 bytes times this length. If the neighborhood is empty, NONE is set to indicate a buffer was not allocated. The total length of the buffer allocated is the buffer for the points plus the length of a Pallet image and a Pallet item header. This space is reserved with an SVC 7. IPNT points to the address in the buffer into which the data should be read.

G D D

PROGRAM SUMMARY SHEET

1) ROUTINE: ATOFFS   2) MODULE: MODE   3) MACHINE: I-4

4) CALLING STATEMENT:

CALL ATOFFS(MSG)

5) ARGUMENTS:

MSG - 8 element cursor status array sent by Pallet.

6) CALLED BY:

Pallet when normal function key is hit

7) CALLS ROUTINES:

NA

8) COMMONS REFERENCED:

CURSTA, MNUTIA

9) PURPOSE AND METHOD:

ATOFFS sets the MODE variable in the CURSTA common to indicate that the system is in NORMAL mode.

G D D

PROGRAM SUMMARY SHEET

1) ROUTINE: AUTOFZ

2) MODULE: DATA EXCEPTION

3) MACHINE: I-4

4) CALLING STATEMENT:

CALL AUTOFZ

5) ARGUMENTS:

NA

6) CALLED BY:

ZMTRNS

7) CALLS ROUTINES:

CLEVEL

8) COMMONS REFERENCED:

COMMUN, DATBAS, DATSTA, MNUTIA

9) PURPOSE AND METHOD:

AUTOFZ determines which features should be displayed, deleted or have a change of detail level after a zoom when the system is in normal mode. For each data base available to the system that has been selected by the user, CLEVEL is called to determine the detail level at which it should be displayed. If this level is different from the current level SELECT is set equal to this level and the DELETE flag is turned on.

G D D

PROGRAM SUMMARY SHEET

1) ROUTINE: AUTONS   2) MODULE: MODE   3) MACHINE: I-4

4) CALLING STATEMENT:

CALL AUTONS(MSG)

5) ARGUMENTS:

MSG - 8 element cursor status array sent by Pallet

6) CALLED BY:

Pallet when the automatic function key is hit

7) CALLS ROUTINES:

NA

8) COMMONS REFERENCED:

CURSTA, MNUTIA

9) PURPOSE AND METHOD:

AUTONS sets the system mode to automatic by changing the MODE variable.

G D D

PROGRAM SUMMARY SHEET

1) ROUTINE: AUTONZ

2) MODULE: DATA EXCEPTION

3) MACHINE: I-4

4) CALLING STATEMENT:

CALL AUTONZ

5) ARGUMENTS:

NA

6) CALLED BY:

ZMTRNS

7) CALLS ROUTINES:

CLEVEL

8) COMMONS REFERENCED:

COMMUN, DATBAS, DATSTA, MNUTIA

9) PURPOSE AND METHOD:

AUTONZ determines which data bases should be displayed, deleted or have a detail level change after a zoom when the system is in automatic mode. For each data base available to the system, CLEVEL is called to determine the proper detail level. If the returned level is not the current level, SELECT is set equal to the returned level, and the DELETE flag is turned on. In addition the AUTOFS flag for that data base is turned on indicating that it is to be considered a user selected data base if the mode is changed to normal.

G D D

PROGRAM SUMMARY SHEET

1) ROUTINE: CLEVEL   2) MODULE: MENU
DATA EXCEPTION   3) MACHINE: I-4

4) CALLING STATEMENT:
CALL CLEVEL(I, LEVEL)

5) ARGUMENTS: I - index into data bases
LEVEL - returns level at which data base should be displayed

6) CALLED BY: MESLCT of MENU module; AUTONZ, AUTOFZ, of DATA EXCEPTION module

7) CALLS ROUTINES:
NA

8) COMMONS REFERENCED:
CURSTA, DATBAS, DATSTA, MNUTIA

9) PURPOSE AND METHOD:

For the current displayed extent, CLEVEL determines the detail level at which the Ith data base should be displayed. LEVEL is set to 0 on entry. If the current extent is not within range CLEVEL returns. If a data base has only one level of detail and falls within range of both the zoom out and zoom in thresholds for that level LEVEL = 1 and CLEVEL returns. The old and new extent values are compared to determine whether a zoom in or zoom out has been done. If a zoom in was done, CLEVEL loops through the zoom in threshold values in order until it finds the level whose threshold is greater than the current extent. LEVEL = 0 if none are greater. For a zoom out, the zoom out thresholds are examined in reverse order starting with the highest detail level. LEVEL is set to the first detail level whose threshold is less than the current extent.

G D D

PROGRAM SUMMARY SHEET

1) ROUTINE: CHARLV	2) MODULE: MENU	3) MACHINE: I-4

4) CALLING STATEMENT:

CALL CHARLV(I,ICHAR)

5) ARGUMENTS: I-feature data base index,ICHAR - a 2 character string returned by CHARLV

6) CALLED BY:

MENUUP, MESLCT

7) CALLS ROUTINES:

NA

8) COMMONS REFERENCED:

COMMUN, DATBAS, DATSTA, MNUTIA

9) PURPOSE AND METHOD:

For a given data base, CHARLV returns a two character string representing the number of the detail level at which the Ith data base is currently displayed, or will be displayed when the menu is entered.

G D D

PROGRAM SUMMARY SHEET

1) ROUTINE: CLMERS   2) MODULE: DATA BASE MANAGEMENT   3) MACHINE: I-4

4) CALLING STATEMENT:

CALL CLMERS(I,ISTART,IEND)

5) ARGUMENTS: I - data base index; ISTART - first column of neighborhood to be erased(1,2,3 or 4);IEND - last column of neighborhood to be erased (1,2,3 or 4)

6) CALLED BY:

MDISP, RPCOL

7) CALLS ROUTINES:

NAME of GDD   ERASE of Pallet

8) COMMONS REFERENCED:

DATBAS, DATSTA, ERASE,TREES

9) PURPOSE AND METHOD:

CLMERS enters the Pallet names of columns of neighborhoods to be erased into the ERSAR array. If the ERSAR array is filled, CLMERS calls Pallet to erase the entries already made. For the data base specified by I, CLMERS constructs the name of any or all columns of the data base and enters them into the ERSAR array. Which columns are entered is determined by ISTART and IEND. There are four columns in a neighborhood; the block number of the head of each column is stored in the COL array in order from left to right. The range of values for ISTART and IEND is 1 to 4. All columns of a neighborhood between and inclusive of ISTART and IEND are erased. If the column represented by an element of the COL array is erased, that element of COL is set to 0.

G D D

PROGRAM SUMMARY SHEET

1) ROUTINE: CRTOMP   2) MODULE: FUNCTION REQUEST   3) MACHINE: I-70

4) CALLING STATEMENT:

CALL CRTOMP(XCURA, YCURA, XCURM, YCURM)

5) ARGUMENTS: XCURA - x absolute position of cursor: YCURA - y absolute position of cursor; XCURM, YCURM - returned map coordinates of cursor.

6) CALLED BY:

SETMSG

7) CALLS ROUTINES:

NA

8) COMMONS REFERENCED:

STATUS

9) PURPOSE AND METHOD:

CRTOMP translates the absolute position of the cursor on the screen to its position on the displayed map in the map coordinate system. It calculates the distance the cursor is from the absolute center of the display. This distance is scaled by the previous extent value and added to the previous center point. (The previous extent and center are used since the cursor was positioned by the user before the translate or scale he requested was done.)

G D D

PROGRAM SUMMARY SHEET

1) ROUTINE: CURPOS 2) MODULE: MENU 3) MACHINE: I-4

4) CALLING STATEMENT:

CALL CURPOS(IX, IY, IDB, IACT)

5) ARGUMENTS: IX- X-position of cursor in absolute coordinates:IY-Y-position of cursor in absolute coordinates; IDB-macro expansion index returned by CURPOS; IACT-function returned by CURPOS

6) CALLED BY:

MESLCT

7) CALLS ROUTINES:

NA

8) COMMONS REFERENCED:

MACRO, MENCON, MNUTIA

9) PURPOSE AND METHOD:

CURPOS determines which line of the menu the cursor is opposite and returns this in IDB; it also determines which function the cursor is under and returns this in IACT. To find out which line the cursor is opposite, the top and bottom coordinates of each line are compared to IY. The line into which IY falls becomes the index into the FETPOS array. For each line on the screen FETPOS contains the index into the macro expansion arrays. IDB is set equal to this index. To determine which function the cursor is under, IX is compared to the x-coordinates of the left and right side of each column. If IX does not fall into a column, IACT = -1. Otherwise it equals ON or OFF.

G D D

PROGRAM SUMMARY SHEET

1) ROUTINE: CURSTA

2) MODULE: DATA EXCEPTION

3) MACHINE: I-4

4) CALLING STATEMENT:

CALL CURSTA(NAME,TYPE,LENGTH,STAT)

5) ARGUMENTS: NAME,TYPE,LENGTH - name of routine to receive the message, type of message and length of message in bytes.
STAT - current display status array

6) CALLED BY:

TRANTP, ZOMTOP via MP

7) CALLS ROUTINES:

ZMTRNS

8) COMMONS REFERENCED:

CURSTA

9) PURPOSE AND METHOD:

CURSTA copies the new current display status array sent by the Function Request module into the CURSTA common. Thus, both the I-4 and I-70 now have the current values for the center point and extent. After copying the new values, CURSTA calls ZMTRNS to test for data exception conditions caused by either a translate or zoom.

G D D

PROGRAM SUMMARY SHEET

1) ROUTINE: ERMSG

2) MODULE: DATA BASE MANAGEMENT

3) MACHINE: I-4

4) CALLING STATEMENT:

CALL(MSG,LEN,NUM)

5) ARGUMENTS: MSG - message to be printed; LEN - number of characters in message; NUM - integer to be printed with message

6) CALLED BY:

RETREV,MSEND

7) CALLS ROUTINES:

NA

8) COMMONS REFERENCED:

NA

9) PURPOSE AND METHOD:

ERMSG prints error messages to the teletype. MSG is moved to an output buffer. NUM is converted to ASCII and also stored in the output buffer. The buffer is printed by an SVC call.

G D D

PROGRAM SUMMARY SHEET

1) ROUTINE: GRDCEN

2) MODULE: DATA EXCEPTION
DATA BASE
MANAGEMENT

3) MACHINE: I-4

4) CALLING STATEMENT:

CALL GRDCEN(I,LEV,NGX,NGY)

5) ARGUMENTS: I - data base index
LEV - level for which neighborhood is being defined
NGX,NGY - coordinates of grid point closest to center point of display.

6) CALLED BY: MTRANS in DATA EXCEPTION
MDISP in DATA BASE MANAGEMENT

7) CALLS ROUTINES:

NA

8) COMMONS REFERENCED:

CURSTA, DATBAS, DATSTA, MAP

9) PURPOSE AND METHOD:

GRDCEN calculates the grid point of a given data base at a given level that is closest to the center point of the display. The X and Y coordinates of the grid point are calculated in a similar manner: for a given data base and level, the width of the blocks into which it is divided is known. The required grid point is the grid point that is no more than half this distance away from the center point. So, the distance between the edge of the entire map and the center point is calculated and then increased by half a block width. This quantity is divided by a block width. The correct coordinate is the quotient; forget the remainder. The coordinate is then checked to be sure it is no less than 2 blocks from an edge. If it is, it is changed so that the neighborhood it defines does not fall outside the mapped area.

G D D

PROGRAM SUMMARY SHEET

1) ROUTINE: IMPTAB   2) MODULE: INITIALIZATION   3) MACHINE: I-70
I-4

4) CALLING STATEMENT:

NA

5) ARGUMENTS:

NA

6) CALLED BY:

IMPINT of MP

7) CALLS ROUTINES:

NA

8) COMMONS REFERENCED:

NA

9) PURPOSE AND METHOD:

IMPTAB is the table which tells MP which routines are to receive which message types. There are two such tables for the GDD; one to be linked on the 70 and one to be linked on the 4.

G D D

PROGRAM SUMMARY SHEET

1) ROUTINE: INIT 2) MODULE: INITIALIZATION 3) MACHINE: I-4

4) CALLING STATEMENT:

NA

5) ARGUMENTS:

NA

6) CALLED BY: operating system start command as the entry point into the GDD

7) CALLS ROUTINES: REDCOM, MDISP of GDD; DRMBFA, ON, CLEAR, DISPLAY, FIND of Pallet; IMPINT, SEND, MP of MP; SETSAV, CHKSAV, INTFMP of FMP

8) COMMONS REFERENCED: COLORS, COMMUN, CURSTA, DATBAS, DATSTA, ERASE, FAC, FILE, MACRO, MAP, MENCON, MNUTIA, TREES

9) PURPOSE AND METHOD:

INIT initializes Pallet, MP and FMP. Using the Pallet ON routine it assigns function buttons to the routines that should be invoked when that button is pushed. It sends a message containing the initial center point and extent, magnification factors and image names to the STATIN routine residing on the I-70 to initialize that side of the GDD. It then displays an empty "world" image to set the proper coordinate system in Pallet, sets the current pointer, and calls MDISP to display the initial data bases specified by the initial value of the COMMUN common. Finally MP is called to wait for messages.

G D D

PROGRAM SUMMARY SHEET

1) ROUTINE: INMVE	2) MODULE: DATA BASE MANAGEMENT	3) MACHINE: I-4

4) CALLING STATEMENT:

CALL INMVE(BUF(IPOS), INDEX)

5) ARGUMENTS: BUF (IPOS) Pointer to index entry read from drum; INDEX returned unpacked index entries

6) CALLED BY:

RINDEX

7) CALLS ROUTINES:

NA

8) COMMONS REFERENCED:

NA

9) PURPOSE AND METHOD:

INMVE unpacks four index entries as they are stored on drum into a FORTRAN integer array. For each index entry, a halfword containing the drum address and a halfword containing the length are required. These are packed into a fullword on drum. INMVE unpacks each halfword into a FORTRAN integer.

G D D

PROGRAM SUMMARY SHEET

1) ROUTINE: MDISP   2) MODULE: DATA BASE MANAGEMENT   3) MACHINE: I-4

4) CALLING STATEMENT:

CALL MDISP(IFORCE)

5) ARGUMENTS:

IFORCE - initially 0; returned as a 1 if MDISP has changed the display

6) CALLED BY: MENUUP of MENU module; ZMTRNS,MTRANS of DATA EXCEPTION module,INIT of INITIALIZATION module

7) CALLS ROUTINES: CLMERS, GRDCEN, SETINX, REDSND, DEALOC of GDD, ERASE of Pallet

8) COMMONS REFERENCED:

COMMUN, DATBAS, DATSTA, ERASE, MNUTIA, TREES

9) PURPOSE AND METHOD:

MDISP erases and displays neighborhoods of feature data bases as dictated by settings of the SELECT and DELETE arrays. MDISP first runs through the DELETE array and makes an entry in the ERSAR array for each column of each neighborhood that is flagged for deletion. Pallet is then called to erase the data from the display tree. Now the SELECT array is examined. Each non-zero entry in the SELECT array is the level at which a data base should be displayed. GRDCEN calculates the grid center of the neighborhood. SETINX reads the index for the neighborhood. For each column, REDSND is called to read the column of data from drum and send it to Pallet.

G D D

PROGRAM SUMMARY SHEET

1) ROUTINE: DBPOS	2) MODULE: MENU	3) MACHINE: I-4

4) CALLING STATEMENT:

CALL DBPOS(I,Y)

5) ARGUMENTS: I - data base index
Y - returned value of Y coordinate of the Itn data base in menu.

6) CALLED BY:

MENUUP, WRTCHR

7) CALLS ROUTINES:

NA

8) COMMONS REFERENCED:

DATBAS, MACRO, MNUTIA

9) PURPOSE AND METHOD:

Given a specific data base, the Ith data base, DBPOS returns the Y coordinate in the map coordinate system, of the line in the menu that represents that data base. The array POSFET has, for each data base, the line number of that feature data base on the screen. By multiplying this line number by 24 (24 dots in the Y axis of a character matrix) and subtracting it from 480 (the Y coordinate of the top of the screen) the Y coordinate of the line of the feature data base is calculated.

G D D

PROGRAM SUMMARY SHEET

1) ROUTINE: MENUUP   2) MODULE: MENU   3) MACHINE: I-4

4) CALLING STATEMENT:

MENUUP(MSG)

5) ARGUMENTS: MSG - eight element cursor status array sent by Pallet when a routine is invoked by a Pallet ON condition.

6) CALLED BY:

Pallet ON condition when menu function key is hit.

7) CALLS ROUTINES: CHARLV, DBPOS, NAME, CHAR, SETSTA, MDISP of GDD; DISPLY, OPENI, CHAR, CLEAR, REFRSH of Pallet

8) COMMONS REFERENCED: COLORS, CURSTA, DATBAS, DATSTA, MENCON, MNUTIA, TREES

9) PURPOSE AND METHOD:

MENUUP displays the menu image and creates and displays an image telling the status of the system and feature data bases. If the menu is already being displayed when MENUUP is invoked, the MDISP routine is called to process the user's menu requests.

G D D

PROGRAM SUMMARY SHEET

1) ROUTINE: MESLCT 2) MODULE: MENU 3) MACHINE: I-4

4) CALLING STATEMENT:
CALL MESLCT(MSG)

5) ARGUMENTS: MSG - eight element cursor status array sent by Pallet when a routine is invoked by a Pallet ON condition.

6) CALLED BY:
Pallet ON condition when select function button is hit

7) CALLS ROUTINES:
CURPOS, RESPON, SETCHR, CLEVEL, CHARLV

8) COMMONS REFERENCED: COLORS, COMMUN, DATBAS, DATSTA, MACRO, MENCON, MNUTIA

9) PURPOSE AND METHOD:
MESLCT determines which feature in the menu has been selected by the user. It determines which function was requested and, after testing for error conditions, makes the proper entries into the SELECT and DELETE arrays of the COMMUN common. The CURPOS routine is first called to calculate the feature and function requested. An error message is displayed if the cursor is not properly aligned with one or the other. The feature is expanded to a list of features via the macro capability. In the case of an ON function, for each feature in the expansion, CLEVEL is called to calculate the proper detail level for the current extent. This is placed in the SELECT array. In the case of an OFF function, error conditions are checked, and DELETE set non-zero for each feature in the macro expansion.

G D D

PROGRAM SUMMARY SHEET

1) ROUTINE: MSEND	2) MODULE: DATA BASE MANAGEMENT	3) MACHINE: I-4

4) CALLING STATEMENT:

CALL MSEND(NAME,P,TYPE,BUF,LENGTH,ERROR)

5) ARGUMENTS: NAME-name of routine to receive msg; P-priority of msg TYPE-message type; BUF-address of msg; LENGTH-length of col. of data ERROR-MP error return code

6) CALLED BY:

REDSND

7) CALLS ROUTINES:

SEND of MP ERMSG of Pallet

8) COMMONS REFERENCED:

NA

9) PURPOSE AND METHOD:

MSEND sets up the calling sequence to MP and calls MP. It is written in assembly to allow the proper calculation of the length of the buffer containing a column of data. The NAME, PRIORITY, TYPE, BUFFER address are copied into the SEND parameter block. The length of the message is then calculated from the index entry for the column being sent and the length of the headers. SEND is then called and error conditions tested for.

G D D

PROGRAM SUMMARY SHEET

1) ROUTINE: MTRANS

2) MODULE: DATA EXCEPTION

3) MACHINE: I-4

4) CALLING STATEMENT:

CALL MTRANS(IFORCE)

5) ARGUMENTS: IFORCE - initially set to 0, it is set to 1 by the routine MDISP if the display has been changed.

6) CALLED BY:

ZMTRNS

7) CALLS ROUTINES:

GRDCEN, MDISP, RPCOL

8) COMMONS REFERENCED:

COMMUN, DATBAS, DATSTA, MNUTIA

9) PURPOSE AND METHOD:

MTRANS determines which data bases need a new neighborhood due to a translation of the center point. It either calls for the replacement of an entire neighborhood or simply one or two columns of the neighborhood. For each data base that is currently displayed, GRDCEN is called to determine the (X,Y) coordinates of the grid point closest to the center point. These X,Y coordinates are compared to the old value. If the Y coordinates are different, SELECT is set equal to the current level and the DELETE flag turned on to force a neighborhood change. If only the X coordinate is different RPCOL is called to change only one or two columns of the neighborhood.

G D D

PROGRAM SUMMARY SHEET

1) ROUTINE: NAME	2) MODULE: MENU DATA BASE MANAGEMENT	3) MACHINE: I-4

4) CALLING STATEMENT:

CALL NAME(FIRST, SEC,RNAME)

5) ARGUMENTS:

FIRST - 1st four characters of name
SEC - 2nd four characters of name
RNAME - 8 character name returned

6) CALLED BY: MENUUP, WRTCHR of MENU module; CLMERS, REDSND of DBM module

7) CALLS ROUTINES:

NA

8) COMMONS REFERENCED:

NA

9) PURPOSE AND METHOD:

NAME constructs an 8-character Pallet name from two four character strings. The two strings are simply concatenated and returned in the RNAME argument which must be of dimension 2 in the calling program.

G D D

PROGRAM SUMMARY SHEET

1) ROUTINE: NEWCEN

2) MODULE: FUNCTION REQUEST

3) MACHINE: I-70

4) CALLING STATEMENT:

CALL NEWCEN(FAC,OXCENM, OYCENM, XCENM, YCENM)

5) ARGUMENTS: FAC - magnification factor

OXCENM,OYCENM - previous center of map in map coordinates

XCENM,YCENM - returned new center in map coordinates

6) CALLED BY:

ZOMTOP

7) CALLS ROUTINES:

NA

8) COMMONS REFERENCED:

STATUS

9) PURPOSE AND METHOD:

NEWCEN calculates the new center of the displayed map when a zoom is requested. Since the point designated by the cursor remains stationary when a zoom is done, there is an implied translate in any zoom. The new center is calculated as the difference between the cursor position in map coordinates and the difference between the cursor position and old center point multiplied by the magnification factor, i.e., XCURM - (XCURM - OXCENM) *FAC

G D D

PROGRAM SUMMARY SHEET

1) ROUTINE: REDCOM 2) MODULE: INITIALIZATION 3) MACHINE: I-4

4) CALLING STATEMENT:

CALL REDCOM

5) ARGUMENTS:

NA

6) CALLED BY:

INIT

7) CALLS ROUTINES:

NA

8) COMMONS REFERENCED:

NA

9) PURPOSE AND METHOD:

REDCOM reads the initial values of all common variables into core from the tape created by the stand alone program, SETUP. It first rewinds the tape on logical unit 6, and reads a 4 byte record containing the address of the first common location. This address is then used as a parameter to the next SVC tape read which reads the next record into core starting at the address in the first record.

G D D

PROGRAM SUMMARY SHEET

1) ROUTINE: REDSND

2) MODULE: DATA BASE MANAGEMENT

3) MACHINE: I-4

4) CALLING STATEMENT:

CALL REDSND(I,LEVEL,INDEX,IPNT,ICOL)

5) ARGUMENTS: I - data base index ; LEVEL - detail level to be displayed; INDEX - index data; IPNT - points to core buffer; ICOL - column of neighborhood to be displayed

6) CALLED BY:

MDISP, RPCOL

7) CALLS ROUTINES:

MSEND, NAME, RETREV, SETBF, SETITM

8) COMMONS REFERENCED:

DATBAS, DATSTA, TREES

9) PURPOSE AND METHOD:

REDSND reads a column of data from drum, sets up Pallet headers for the data and sends it to Pallet. RETREV is called to read the ICOL column of data into core starting at location IPNT. SETBF and SETITM add image and item headers required by Pallet to the data. MSEND transmits the data to the I-70 using MP.

G D D

PROGRAM SUMMARY SHEET

1) ROUTINE: RESPON 2) MODULE: MENU 3) MACHINE: I-4

4) CALLING STATEMENT:

CALL RESPON(ICHAR,LEN,ICOLOR)

5) ARGUMENTS: ICHAR - character string
LEN - length of character string
ICOLOR - color of character string

6) CALLED BY:

MESLCT

7) CALLS ROUTINES:

ERASE, OPENI, CHAR, DISPLY of Pallet

8) COMMONS REFERENCED:

MENCON, MNUTIA, TREES

9) PURPOSE AND METHOD:

RESPON is used to display responses to the user whenever a menu function has been requested. It first erases the old response image, and then opens a new one. The ICHAR character string is placed in this image. The image is then displayed by attaching it to the status image.

G D D

PROGRAM SUMMARY SHEET

1) ROUTINE: RETREV   2) MODULE: DATA BASE MANAGEMENT   3) MACHINE: I-4

4) CALLING STATEMENT:

CALL RETREV(IFILE,INDEX(I,ICOL),IPNT,NONE)

5) ARGUMENTS: IFILE-data base file number; INDEX(I,ICOL)-index information for ICOLth column of data base; IPNT-address into which data should be read; NONE-returned flag set non zero if column is empty

6) CALLED BY:

REDSND

7) CALLS ROUTINES:

ERMSG

8) COMMONS REFERENCED:

NA

9) PURPOSE AND METHOD:

RETREV reads a column of data from drum into core. The block address of the ICOL column of data is taken from the index and used by the DRUM utility to find the data on drum. DRUM reads the number of points specified by the index. Once read, the starting location of the data within the first block read from the drum is calculated. (The low order four bits of the length entry in an index entry identifies which point in the drum block is the first point for the column of data read.) IPNT is returned as this location minus the 24 byte header required by PALLET. A -1 is stored at the end of the data as a PALLET delimiter.

G D D

PROGRAM SUMMARY SHEET

1) ROUTINE: RINDEX	2) MODULE: DATA BASE MANAGEMENT	3) MACHINE: I-4

4) CALLING STATEMENT:

CALL RINDEX(I,LFTTOP,LEV,INDEX)

5) ARGUMENTS: I - data base index; LFTTOP - block number of left top block of neighborhood; LEV - level at which data base is to be displayed INDEX - returned index value for neighborhood

6) CALLED BY:

SETINX

7) CALLS ROUTINES:

ERMSG, INMVE of GDD; DRUM - FORTRAN utility

8) COMMONS REFERENCED:

DATBAS, DATSTA

9) PURPOSE AND METHOD:

RINDEX reads the index entries for each of the four columns of a neighborhood having LFTTOP as the top left block. Each index entry is a fixed 4 bytes long. The block address in the index file of a specific index entry is four times LFTTOP divided by 128 bytes per block. In case the entry wanted is at the very end of the calculated block, over a block is read to insure all four entries are read into core. The actual byte position in the block is calculated as an index in a FORTRAN array. INMVE is called to unpack the index data into a FORTRAN integer array.

G D D

PROGRAM SUMMARY SHEET

1) ROUTINE: RPCOL

2) MODULE: DATA BASE MANAGEMENT

3) MACHINE: I-4

4) CALLING STATEMENT:

CALL RPCOL(I,NGX,NGY,IFORCE)

5) ARGUMENTS: I - data base index; NGX,NGY - coordinates of new grid center IFORCE - set to 1 if RPCOL changes display

6) CALLED BY:

MTRANS

7) CALLS ROUTINES:

CLMERS, SETINX, REDSND, DEALOC

8) COMMONS REFERENCED:

DATBAS, DATSTA, ERASE, TREES

9) PURPOSE AND METHOD:

RPCOL erases and displays partial neighborhoods when a translation does not require an entirely new neighborhood. It first calculates which one or two columns need to be erased as a function of the difference between the old and new X grid coordinate. CLMERS make the necessary entries into the ERSAR array. Next, the entries in the COL array are rotated to maintain the left to right order of the columns in the data base. Those elements of COL cleared by the rotate will be filled by the block numbers of the new columns and order will be maintained. SETINX is called to retrieve the index for the new neighborhood. For each column that needs to be displayed, REDSND reads the data and sends it to Pallet.

G D D

PROGRAM SUMMARY SHEET

1) ROUTINE: SETBF	2) MODULE: DATA BASE MANAGEMENT	3) MACHINE: I-4

4) CALLING STATEMENT:

CALL SETBF(TREE,WHERE,REFRSH,IPNT)

5) ARGUMENTS: TREE - Pallet device to which image should be attached ; WHERE-name of the node to which image should be attached; REFRSH-refresh type IPNT-points to where header should be stored.

6) CALLED BY:

REDSND

7) CALLS ROUTINES:

NA

8) COMMONS REFERENCED:

NA

9) PURPOSE AND METHOD:

SETBF creates a Pallet image header for a column of data about to be sent to Pallet for display. The TREE, WHERE and REFRSH parameters are stored in order in successive locations starting at address IPNT. IPNT is returned pointing to the next free location.

G D D

PROGRAM SUMMARY SHEET

1) ROUTINE: SETINX   2) MODULE: DATA BASE MANAGEMENT   3) MACHINE: I-4

4) CALLING STATEMENT:

CALL SETINX(I,LEVEL,NGX,NGY,LFTTOP,INDEX,IPNT,NONE)

5) ARGUMENTS: I - data base index; LEVEL - level for display; NGX,NGY - grid center coordinates; LFTTOP - returned block number of left top block in neighborhood; IPNT-returned address of core block allocated for data; NONE - returned flag

6) CALLED BY:

MDISP, RPCOL

7) CALLS ROUTINES:

TOPLFT, RINDEX, ALLOC

8) COMMONS REFERENCED:

NA

9) PURPOSE AND METHOD:

SETINX sets up the retrieval of a neighborhood of columns from the drum. It calculates the proper entry into the index, reads the index and then allocates a core buffer into which a column of data can be read. NONE is set non-zero if the neighborhood contains no data.

G D D

PROGRAM SUMMARY SHEET

1) ROUTINE: SETITM	2) MODULE: DATA BASE MANAGEMENT	3) MACHINE: I-4

4) CALLING STATEMENT:

CALL SETITM(ITYPE,N,COLOR,NAME,IPNT)

5) ARGUMENTS: ITYPE-data type, 1 points, 2 lines; N-number of points in item COLOR-color of item; NAME-name of ITEM

6) CALLED BY:

REDSND

7) CALLS ROUTINES:

NA

8) COMMONS REFERENCED:

NA

9) PURPOSE AND METHOD:

SETITM constructs an item header for a column of data about to be displayed. ITYPE, N, COLOR and NAME are stored in successive locations starting at location IPNT. This item header immediately follows the image header and immediately precedes the data itself.

G D D

PROGRAM SUMMARY SHEET

1) ROUTINE: SETMSG	2) MODULE: FUNCTION REQUEST	3) MACHINE: I-70

4) CALLING STATEMENT:

CALL SETMSG(MSG)

5) ARGUMENTS: MSG - 8 element cursor status array sent by PALLET when a function key is hit.

6) CALLED BY:

ZOMTOP, TRANTP

7) CALLS ROUTINES:

CRTOMP

8) COMMONS REFERENCED:

STATUS

9) PURPOSE AND METHOD:

SETMSG sets the display status array, CURSTA, for shipment to the Data Exception module. The CURSTA array contains 16 elements, 8 for current values and 8 for previous values. When called, SETMSG copies the current values to the previous value locations. The new absolute cursor position is set in CURSTA, converted to map coordinates and also stored in CURSTA. (The new center point and extent are set by the ZOMTOP and TRANTP routines after calling SETMSG.)

G D D

PROGRAM SUMMARY SHEET

1) ROUTINE: SETSTA	2) MODULE: MENU	3) MACHINE: I-4
4) CALLING STATEMENT: CALL SETSTA(ICHAR,LEN)		
5) ARGUMENTS: ICHAR - character string to be displayed in status line of menu ; LEN - length of character string		
6) CALLED BY: MENUUP		
7) CALLS ROUTINES: CHAR of Pallet		
8) COMMONS REFERENCED: COLORS, MENCON		
9) PURPOSE AND METHOD: SETSTA enters the character string ICHAR into the status image using the CHAR command of Pallet.		

G D D

PROGRAM SUMMARY SHEET

1) ROUTINE: STATCS 2) MODULE: MODE 3) MACHINE: I-4

4) CALLING STATEMENT:

CALL STATCS (MSG)

5) ARGUMENTS:

MSG - 8 element cursor status array sent by Pallet

6) CALLED BY:

Pallet when special mode function key is hit.

7) CALLS ROUTINES:

NA

8) COMMONS REFERENCED:

MNUTIA, CURSTA

9) PURPOSE AND METHOD:

STATCS changes the system mode to special by setting the MODE variable.

G D D

PROGRAM SUMMARY SHEET

1) ROUTINE: STATIN   2) MODULE: INITIALIZATION   3) MACHINE: I-70

4) CALLING STATEMENT:

CALL STATIN (NAME, TYPE, LENGTH, STAT)

5) ARGUMENTS: NAME, TYPE, LENGTH - name of routine to receive the message, type of message and length of message in bytes.
STAT - initial values for the STATUS and DATA commons.

6) CALLED BY:

INIT via MP

7) CALLS ROUTINES:

NA

8) COMMONS REFERENCED:

DATA, STATUS

9) PURPOSE AND METHOD:

STATIN initializes the two commons, DATA and STATUS, that reside on the I-70. The array STAT is filled with the proper information by INIT, which sends it via MP to STATIN. STATIN then copies the values received into the two common areas.

G D D

PROGRAM SUMMARY SHEET

1) ROUTINE: TOPLFT	2) MODULE: DATA BASE MANAGEMENT	3) MACHINE: I-4

4) CALLING STATEMENT:

CALL TOPLFT(I,LEVEL,NGX,NGY,LFTTOP)

5) ARGUMENTS: I - data base index; LEVEL - detail level of data base; NGX,NGY-grid point center of neighborhood; LFTTOP - returned value of number of top left block in neighborhood

6) CALLED BY:

SETINX

7) CALLS ROUTINES:

NA

8) COMMONS REFERENCED:

DATBAS, DATSTA

9) PURPOSE AND METHOD:

TOPLFT calculates the block number of the top left block of a sixteen block neighborhood having a grid point center of (NGX,NGY). The block number is returned as if the blocks were numbered from left to right starting in the bottom row.

G D D

PROGRAM SUMMARY SHEET

1) ROUTINE: TRANTP	2) MODULE: FUNCTION REQUEST	3) MACHINE: I-70

4) CALLING STATEMENT:

CALL TRANTP(MSG)

5) ARGUMENTS:

MSG - 8 element cursor status array sent by Pallet

6) CALLED BY:

Pallet ON condition when translate function key is hit.

7) CALLS ROUTINES: SETMSG of GDD; TRANS of Pallet; SEND of MP

8) COMMONS REFERENCED:

DATA, STATUS

9) PURPOSE AND METHOD:

TRANTP requests an immediate translate of the available neighborhood, sets the new center of display in the CURSTA array, and sends the CURSTA array to the Data Exception module.

G D D

PROGRAM SUMMARY SHEET

1) ROUTINE: WRTCHR   2) MODULE: MENU   3) MACHINE: I-4

4) CALLING STATEMENT:

CALL WRTCHR(I,X,SECNME, ICHAR, ICOLR)

5) ARGUMENTS: I-index of data base opposite which character is to be written; X-x coordinate in menu coordinates where character should appear;SECNME-second half of name of character string;ICHAR-character to be written;ICOLR-color of character

6) CALLED BY:

MESLCT

7) CALLS ROUTINES:

DBPOS, NAME of GDD; OPENI, CHAR, DISPLY of Pallet

8) COMMONS REFERENCED:

DATBAS, DATSTA, MENCON, MNUTIA, TREES

9) PURPOSE AND METHOD:

WRTCHR writes a two character string to the display opposite the line in the menu that represents the Ith data base. DBPOS is called and returns the Y coordinate in the menu coordinate system of the proper line in the menu. A pallet name for the character string to be written is constructed by NAME from the PREFIX for the Ith data base and the 4 characters in the SECNME argument. An image is opened with this name, the ICHAR string placed in it and displayed.

G D D

PROGRAM SUMMARY SHEET

1) ROUTINE: ZMINTP	2) MODULE: FUNCTION REQUEST	3) MACHINE: I-70

4) CALLING STATEMENT:

CALL ZMINTP(MSG)

5) ARGUMENTS: MSG - 8 element cursor status array sent by Pallet when a function button is hit.

6) CALLED BY:

Pallet ON condition when zoom in function key is hit.

7) CALLS ROUTINES:

ZOMTOP

8) COMMONS REFERENCED:

DATA

9) PURPOSE AND METHOD:

ZMINTP sets the zoom in magnification factor and calls ZOMTOP to finish processing the zoom function request.

G D D

PROGRAM SUMMARY SHEET

1) ROUTINE: ZMOUTP	2) MODULE: FUNCTION REQUEST	3) MACHINE: I-70

4) CALLING STATEMENT:

CALL ZMOUTP(MSG)

5) ARGUMENTS: MSG - 8 element cursor status array sent by Pallet when a function key is hit.

6) CALLED BY:

Pallet ON condition when zoom out function key is hit.

7) CALLS ROUTINES:

ZOMTOP

8) COMMONS REFERENCED:

DATA

9) PURPOSE AND METHOD:

Sets the zoom out magnification factor and calls ZOMTOP to finish processing the zoom request.

G D D

PROGRAM SUMMARY SHEET

1) ROUTINE: ZMTRNS

2) MODULE: DATA EXCEPTION

3) MACHINE: I-4

4) CALLING STATEMENT:

CALL ZMTRNS

5) ARGUMENTS:

NA

6) CALLED BY:

CURSTA

7) CALLS ROUTINES:

AUTONZ, AUTOFZ, MDISP, MTRANS of GDD; REFRSH of PALLET

8) COMMONS REFERENCED:

CURSTA, MNUTIA, TREES

9) PURPOSE AND METHOD:

ZMTRNS is the top level data exception routine. It decides which of the zoom algorithms should be called as a function of the mode of the system, calls the MTRANS routine to determine any data exceptions caused by a requested or implied translate, and calls the MDISP routine to retrieve new neighborhoods for data bases with data exceptions.

G D D

PROGRAM SUMMARY SHEET

1) ROUTINE: ZOMTOP

2) MODULE: FUNCTION REQUEST

3) MACHINE: I-70

4) CALLING STATEMENT:

CALL ZOMTOP(MSG,FAC)

5) ARGUMENTS: MSG - 8 element cursor status array sent by Pallet
FAC - magnification factor

6) CALLED BY:

ZMINTP, ZMOUTP

7) CALLS ROUTINES: SETMSG, NEWCEN of GDD ; SCALE of Pallet ; SEND of MP

8) COMMONS REFERENCED:

STATUS, DATA

9) PURPOSE AND METHOD:

ZOMTOP requests the immediate scaling of the available neighborhood, calculates the new center point and scale, and sends the new display status array, CURSTA, to the Data Exception module.

APPENDIX VI

PROGRAM LISTINGS

```
0000R                    ENTRY ABIND,AARC,ABLOCK
0000R           ABIND    EQU   *
0000R           AARC     EQU   *
0000R           ABLOCK   EQU   *
0000R 030F               BR    15
0002R                    END
```

```
0000R                  ENTRY ALLOC,DEALOC
                *       CALL  ALOOC(INDEX,IPNT,NONE)
0000R                  EXTRN ERMSG
0000            RET    EQU   0
0002            INDEX  EQU   2
0004            IPNT   EQU   4
0006            NONE   EQU   6
0008            LENGTH EQU   8
0000R           SAVE   DS    32
0020R D000      ALLOC  STM   3,SAVE
      0000R
0024R 482F             LH    INDEX,INDEX(15)
      0002
0028R 484F             LH    IPNT,IPNT(15)
      0004
002CR 486F             LH    NONE,NONE(15)
      0006
0030R C810             LHI   1,3               SET LOOP LIMIT
      0003
0034R 4882             LH    LENGTH,4(INDEX) GET FIRST LENGTH
      0004
0038R CC80             SRHL  LENGTH,4          SHIFT OUT BLOCK LOCATION
      0004
003CR C832             LHI   3,12(INDEX)       GET ADDRES OF NEXT INDEX ENTRY
      000C
0040R 08A8      LOOP   LHR   10,LENGTH         COPY LENGTH INTO SCRATCH REGISTER
0042R 4873             LH    7,0(3)            LOAD NEXT LENGTH
      0000
0046R CC70             SRHL  7,4               SHIFT OUT BLOCK LOCATION
      0004
004AR 0BA7             SHR   10,7              WHICH IS BIGGER
004CR 4310             BNM   LOOP1             CONTINUE IF LENGTH IS BIGGER
      0052R
0050R 0887             LHR   LENGTH,7          SET NEW LENGTH
0052R C833      LOOP1  LHI   3,8(3)            POINT TO NEXT INDEX ENTRY
      0008
0056R CB10             SHI   1,1               DONE LOOPING
      0001
005AR 4230             BNZ   LOOP              LOOP IF NOT DONE
      0040R
005ER CD80             SLHL  LENGTH,3          MULT BY 8 TO GET TOTAL NUMBER OF BYTES
      0003
0062R 4230             BNZ   SVC               TEST FOR EMPTY COLUMN
      006ER
0066R C8A0             LHI   10,1              SET RETURN CODE FOR EMPTY COLUMN
      0001
006AR 4300             B     RETUR             RETURN
      009CR
006ER CA80      SVC    AHI   LENGTH,128+24+6 CALC TOTAL LENGTH NEEDED
      0091
0072R 4080             STH   LENGTH,LENG       STORE LENGTH IN SVC BLOCK
      00E8R
0076R E170             SVC   7,ALOC            RESERVE THE SPACE
      00E4R
007AR 4810             LH    1,ERROR           IS THERE AN ERROR
      00E6R
```

```
ØØ7ER 433Ø          BZ    CONT            SKIP IF NO ERROR
      ØØ8ER
ØØ82R 41FØ          BAL   15,ERMSG
      ØØØØF
ØØ86R ØØØ8          DC    8
ØØ88R ØØAAR         DC    A(MSG1)
ØØ8AR ØØB4R         DC    A(TEN)
ØØ8CR ØØE6R         DC    A(ERROR)
ØØ8ER 481Ø   CONT   LH    1,BUF           GET ADDRES OF RESERVED CORE
      ØØEAR
ØØ92R CA1Ø          AHI   1,24            ALLOW ROOM FOR HEADER
      ØØ18
ØØ96R 4Ø14          STH   1,Ø(IPNT)       STORE IN RETURN VARIABLE
      ØØØØ
ØØ9AR Ø7AA          XHR   1Ø,1Ø           SET RETURN CODE
ØØ9CR 4ØA6   RETUR  STH   1Ø,Ø(NONE)
      ØØØØ
ØØAØR D1Ø4   RETURD LM    Ø,SAVE
      ØØØØR
ØØA4R 4AFF          AH    15,RET(15)
      ØØØØ
ØØA8R Ø3ØF          BR    15
ØØAAR 414C   MSG1   DC    C'ALLOC SVC7'
      4C4F
      432Ø
      5356
      4337
ØØB4R ØØØA   TEN    DC    1Ø
ØØB6R ØØØB   ELVN   DC    11
ØØB8R 4445   MSG2   DC    C'DEALOC SVC7 '
      414C
      4F43
      2Ø53
      5643
      372Ø
ØØC4R DØØØ   DEALOC STM   Ø,SAVE
      ØØØØR
ØØC8R 481Ø          LH    1,LENG
      ØØE8R
ØØCCR 4Ø1Ø          STH   1,LENGD
      Ø1ØØR
ØØDØR 481Ø          LH    1,BUF
      ØØEAR
ØØD4R 4Ø1Ø          STH   1,BUFD
      Ø1Ø2R
ØØD8R E17Ø          SVC   7,DALOC
      ØØFCR
ØØDCR 481Ø          LH    1,DALOC+2
      ØØFER
ØØEØR 433Ø          BZ    RETURD
      ØØAØR
ØØE4R 41FØ          BAL   15,ERMSG
      ØØB4R
ØØE8R ØØØ8          DC    8
ØØEAR ØØB8R         DC    A(MSG2)
ØØECR ØØB6R         DC    A(ELVN)
```

```
ØØEER ØØFER            DC     DALOC+2
ØØFØR 43ØØ             B      RETURD
      ØØAØR
ØØF4R ØØØ3  ALOC       DC     3
ØØF6R ØØØØ  ERROR      DC     Ø
ØØF8R ØØØØ  LENG       DC     *-*
ØØFAR ØØØØ  BUF        DC     *-*
ØØFCR ØØØ4  DALOC      DC     4
ØØFER ØØØØ             DC     Ø
Ø1ØØR ØØØØ  LENGD      DC     *-*
Ø1Ø2R ØØØØ  BUFD       DC     *-*
Ø1Ø4R                  END
```

```
      SUBROUTINE ATOFFS(MSG)
      COMMON /MNUTIA/ ON,OFF,YES,NO,UP,PASS
     1 ,AUTON,AUTOF,STATIC
      COMMON /CURSTA/ MODE,XCENM,YCENM,OXCEN,OYCEN,XEXTNT,YEXTNT
     1,OXXTNT,OYXTNT
     2,XCURA,YCURA,OXCURA,OYCURA,XCURM,YCURM,OXCURM,OYCURM
      INTEGER  ON,OFF,YES,NO,UP,PASS
      INTEGER AUTON,AUTOF,STATIC
      MODE=AUTOF
      RETURN
      END
```

```
      SUBROUTINE AUTOFZ
      COMMON /DATSTA/ CURLEV(10),AUTOFS(10),GX(10),GY(10),COL(4,10)
      COMMON /DATBAS/ PREFIX(10),NUMLEV(10),INMENU(10),ZMOTHR(4,10),
     1ZMINTH(4,10),D(4,10),NUMX(4,10),NUMY(4,10),IFILE(4,10),
     2 DBINDX(4,10),ITYPE(4,10),ICOLOR(4,10),NUMDB
      COMMON /MNUTIA/ ON,OFF,YES,NO,UP,PASS
     1 ,AUTON,AUTOF,STATIC
      COMMON /COMMUN/ SELECT(10),DELETE(10)
      INTEGER SELECT,DELETE
      INTEGER DBINDX
      INTEGER CURLEV,AUTOFS,GX,GY,COL
      INTEGER  ON,OFF,YES,NO,UP,PASS
      INTEGER AUTON,AUTOF,STATIC
      DO 10 I=1,NUMDB
         IF(AUTOFS(I) .EQ. OFF) GO TO 10
            CALL CLEVEL(I,LEVEL)
            IF(LEVEL.EQ. CURLEV(I)) GO TO 10
               SELECT(I) =LEVEL
               DELETE(I) = ON
               IF(CURLEV(I) .EQ. 0) DELETE(I) =OFF
10    CONTINUE
      RETURN
      END
```

```
      SUBROUTINE AUTONS(MSG)
      COMMON /MNUTIA/ ON,OFF,YES,NO,UP,PASS
     1 ,AUTON,AUTOF,STATIC
      COMMON /CURSTA/ MODE,XCENM,YCENM,OXCEN,OYCEN,XEXTNT,YEXTNT
     1,OXXTNT,OYXTNT
     2,XCURA,YCURA,OXCURA,OYCURA,XCURM,YCURM,OXCURM,OYCURM
      INTEGER  ON,OFF,YES,NO,UP,PASS
      INTEGER AUTON,AUTOF,STATIC
      MODE = AUTON
      RETURN
      END
```

```
      SUBROUTINE AUTONZ
      COMMON /MNUTIA/ ON,OFF,YES,NO,UP,PASS
     1 ,AUTON,AUTOF,STATIC
      COMMON /DATSTA/ CURLEV(10),AUTOFS(10),GX(10),GY(10),COL(4,10)
      COMMON /DATBAS/ PREFIX(10),NUMLEV(10),INMENU(10),ZMOTHR(4,10),
     1ZMINTH(4,10),D(4,10),NUMX(4,10),NUMY(4,10),IFILE(4,10),
     2 DBINDX(4,10),ITYPE(4,10),ICOLOR(4,10),NUMDB
      COMMON /COMMUN/ SELECT(10),DELETE(10)
      INTEGER CURLEV,AUTOFS,GX,GY,COL
      INTEGER SELECT,DELETE
      INTEGER  ON,OFF,YES,NO,UP,PASS
      INTEGER AUTON,AUTOF,STATIC
      DO 10 I=1,NUMDB
         CALL CLEVEL(I,LEVEL)
         IF(LEVEL .EQ. CURLEV(I))GO TO 10
            SELECT(I) = LEVEL
            DELETE(I) = ON
            AUTOFS(I) = ON
            IF(LEVEL .EQ. 0) AUTOFS(I) = OFF
            IF(CURLEV(I) .EQ. 0) DELETE(I)=OFF
10    CONTINUE
      RETURN
      END
```

```
      SUBROUTINE CHARLV(I,ICHAR)
      COMMON /MNUTIA/ ON,OFF,YES,NO,UP,PASS
     1 ,AUTON,AUTOF,STATIC
      COMMON /COMMUN/ SELECT(10),DELETE(10)
      COMMON /DATSTA/ CURLEV(10),AUTOFS(10),GX(10),GY(10),COL(4,10)
      COMMON /DATBAS/ PREFIX(10),NUMLEV(10),INMENU(10),ZMOTHR(4,10),
     1ZMINTH(4,10),D(4,10),NUMX(4,10),NUMY(4,10),IFILE(4,10),
     2 DBINDX(4,10),ITYPE(4,10),ICOLOR(4,10),NUMDB
      INTEGER DBINDX
      INTEGER  ON,OFF,YES,NO,UP,PASS
      INTEGER AUTON,AUTOF,STATIC
      INTEGER SELECT,DELETE
      INTEGER CURLEV,AUTOFS,GX,GY,COL
      DIMENSION IASCII(10)
      DATA IASCII(1),IASCII(2),IASCII(3),IASCII(4),IASCII(5),IASCII(6),
     1 IASCII(7),IASCII(8),IASCII(9),IASCII(10)
     2/2H0 ,2H1 ,2H2 ,2H3 ,2H4 ,2H5 ,2H6 ,2H7 ,2H8 ,2H9 /
      LEV = CURLEV(I)
      IF (SELECT(I) .NE.OFF) LEV=SELECT(I)
      ICHAR = IASCII (LEV+1)
      RETURN
      END
```

```
      SUBROUTINE CLEVEL(I,LEVEL)
      COMMON /MNUTIA/ ON,OFF,YES,NO,UP,PASS
     1 ,AUTON,AUTOF,STATIC
      COMMON /CURSTA/ MODE,XCENM,YCENM,OXCEN,OYCEN,XEXTNT,YEXTNT
     1,OXXTNT,OYXTNT
     2,XCURA,YCURA,OXCURA,OYCURA,XCURM,YCURM,OXCURM,OYCURM
      COMMON /DATSTA/ CURLEV(10),AUTOFS(10),GX(10),GY(10),COL(4,10)
      COMMON /DATBAS/ PREFIX(10),NUMLEV(10),INMENU(10),ZMOTHR(4,10),
     1ZMINTH(4,10),D(4,10),NUMX(4,10),NUMY(4,10),IFILE(4,10),
     2 DBINDX(4,10),ITYPE(4,10),ICOLOR(4,10),NUMDB
      INTEGER CURLEV,AUTOFS,GX,GY,COL
      INTEGER DBINDX
      INTEGER  ON,OFF,YES,NO,UP,PASS
      INTEGER AUTON,AUTOF,STATIC
      LEVEL = 0
C TEST FOR OUT OF RANGE
      IF(XEXTNT .GT. ZMOTHR(1,I)) GO TO 60
C PROCESS IF ONLY ONE LEVEL OF DETAIL
      IF(NUMLEV(I) .GT. 1) GO TO 10
         IF ((XEXTNT .LE. ZMOTHR(1,I)) .AND.(XEXTNT .GE. ZMINTH(1,I))
     1) LEVEL = 1
      GO TO 50
C   ZOOM IN OR ZOOM OUT
10    IF(XEXTNT .LE.  OXXTNT)GO TO 30
C MUST HAVE ZOOMED OUT
      LOOP = NUMLEV(I)
      DO 20 J= 1,LOOP
         LEV = LOOP-J + 1
         IF(XEXTNT .GT. ZMOTHR(LEV,I)) GO TO 20
            LEVEL = LEV
            GO TO 50
20    CONTINUE
      GO TO 60
C MUST HAVE ZOOMED IN
30    LOOP = NUMLEV(I)
      DO 40 LEV= 1,LOOP
         IF(XEXTNT .LT. ZMINTH(LEV,I)) GO TO 40
            LEVEL = LEV
            GO TO 50
40    CONTINUE
      GO TO 60
50    IF((XEXTNT .GT. ZMOTHR(LEVEL,I)).OR.(XEXTNT.LT.ZMINTH(LEVEL,I)))
     1  LEVEL = 0
60    RETURN
      END
```

```
      SUBROUTINE CLMERS(I,ISTART,IEND)
      COMMON /ERASE/ ERSAR(2,40),ICNT,ERSIZE
      COMMON /TREES/ MAPTRE,MENTRE,WORLD(2),ZINBUT,TRNBUT,SLCTBT,
     1 AUFBUT,ZOTBUT,MENBUT,AONBUT,STABUT
     2 ,RMAP(2)
      COMMON /DATSTA/ CURLEV(10),AUTOFS(10),GX(10),GY(10),COL(4,10)
      COMMON /DATBAS/ PREFIX(10),NUMLEV(10),INMENU(10),ZMOTHR(4,10),
     1ZMINTH(4,10),D(4,10),NUMX(4,10),NUMY(4,10),IFILE(4,10),
     2 DBINDX(4,10),ITYPE(4,10),ICOLOR(4,10),NUMDB
      INTEGER CURLEV,AUTOFS,GX,GY,COL
      INTEGER DBINDX
      INTEGER ERSIZE
      INTEGER ZINBUT,TRNBUT,SLCTBT ,AUFBUT,ZOTBUT,AONBUT,STABUT
      DO 20 ICOL = ISTART,IEND
         IF(COL(ICOL,I) .EQ. 0) GO TO 20
         IF(ICNT .LT. ERSIZE) GO TO 10
            CALL ERASE (MAPTRE,ERSAR,ICNT,2)
            ICNT = 0
10       ICNT = ICNT+1
         CALL NAME(PREFIX(I) ,COL(ICOL,I),ERSAR(1,ICNT))
         COL(ICOL,I) = 0
20    CONTINUE
      RETURN
      END
```

```
      SUBROUTINE CRTOMP(XCURA,YCURA,XCURM,YCURM)
      COMMON /STATUS/ CURSTA(16)
      INTEGER XCENM,YCENM,XXTNT,YXTNT,XCURB,YCURB,XCURN,YCURN
      INTEGER OXCENM,OYCENM,OXXTNT,OYXTNT,OXCURA,OYCURA,OXCURM,OYCURM
      DATA XCENM,YCENM,XXTNT,YXTNT,XCURB,YCURB,XCURN,YCURN,
     1 OXCENM,OYCENM,OXXTNT,OYXTNT,OXCURA,OYCURA,OXCURM,OYCURM
     2  /1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16/
C   CALCULATE DISTANCE CURSOR IS FROM  CENTER
      ABSX = XCURA -256.
      ABSY = YCURA -240.
C   SCALE THAT DISTANCE AND ADD TO CENTER OF MAP TO GET CURSOR POSITION
C   IN TERMS OF MAP COORDINATES
      XCURM = CURSTA(OXCENM) + ABSX*CURSTA(OXXTNT)
      YCURM = CURSTA(OYCENM) + ABSY * CURSTA(OYXTNT)
      RETURN
      END
```

```
      SUBROUTINE CURPOS(IX,IY,IDB,IACT)
      COMMON /MNUTIA/ ON,OFF,YES,NO,UP,PASS
     1 ,AUTON,AUTOF,STATIC
      COMMON /MENCON/ MENU,ONXC,ONRXC,OFFXC,OFFRXC,MENNME(2),
     1  STATUS(2),SYSTAT(2),SYSRES(2)
     2,STATY,RESYC,RLEFT
      COMMON /MACRO/ MACNUM(10),MACEXP(10,4),FETPOS(10),POSFET(10)
     1 ,NUMFET
      REAL MENNME
      INTEGER FETPOS,POSFET
      INTEGER  ON,OFF,YES,NO,UP,PASS
      INTEGER AUTON,AUTOF,STATIC
C CALCULATE POSITION OF CURSOR IN FEATURES
      IDB = 0
      NUM = NUMFET+3
      DO 10 J = 3,NUM
         K=480-(J-1) *24
         K1=K-24
         IF((IY.LE.K).AND. (IY.GE. K1)) GO TO 15
10    CONTINUE
      GO TO 20
15    IDB = FETPOS(J)
C SET ACTION TO BE TAKEN
20    IACT = -1
      X = FLOAT(IX)
      Y = FLOAT(IY)
      IF (( X .GE. ONXC).AND. (X .LE. ONRXC)) IACT = ON
      IF (( X .GE. OFFXC) .AND. (X .LE. OFFRXC)) IACT = OFF
      RETURN
      END
```

```
      SUBROUTINE CURSTA(NAME,TYPE,LENGTH,STAT)
      COMMON /CURSTA/ MODE,XCENM,YCENM,OXCEN,OYCEN,XEXTNT,YEXTNT
     1,OXXTNT,OYXTNT
     2,XCURA,YCURA,OXCURA,OYCURA,XCURM,YCURM,OXCURM,OYCURM
      REAL NAME
      INTEGER TYPE
      DIMENSION STAT(16)
      XCENM=STAT(1)
      YCENM=STAT(2)
      XEXTNT=STAT(3)
      YEXTNT=STAT(4)
      XCURA = STAT(5)
      YCURA=STAT(6)
      XCURM = STAT(7)
      YCURM= STAT(8)
      OXCEN  = STAT(9)
      OYCEN  = STAT(10)
      OXXTNT = STAT(11)
      OYXTNT = STAT(12)
      OXCURA = STAT(13)
      OYCURA = STAT(14)
      OXCURM = STAT(15)
      OYCURM = STAT(16)
      CALL ZMTRNS
      RETURN
      END
```

```
      SUBROUTINE DBPOS(I,Y)
      COMMON /MACRO/ MACNUM(1Ø),MACEXP(1Ø,4),FETPOS(1Ø),POSFET(1Ø)
     1 ,NUMFET
      COMMON /MNUTIA/ ON,OFF,YES,NO,UP,PASS
     1 ,AUTON,AUTOF,STATIC
      COMMON /DATSTA/ CURLEV(1Ø),AUTOFS(1Ø),GX(1Ø),GY(1Ø),COL(4,1Ø)
      COMMON /DATBAS/ PREFIX(1Ø),NUMLEV(1Ø),INMENU(1Ø),ZMOTHR(4,1Ø),
     1ZMINTH(4,1Ø),D(4,1Ø),NUMX(4,1Ø),NUMY(4,1Ø),IFILE(4,1Ø),
     2 DBINDX(4,1Ø),ITYPE(4,1Ø),ICOLOR(4,1Ø),NUMDB
      INTEGER DBINDX
      INTEGER CURLEV,AUTOFS,GX,GY,COL
      INTEGER FETPOS,POSFET
      INTEGER  ON,OFF,YES,NO,UP,PASS
      INTEGER AUTON,AUTOF,STATIC
      Y=Ø.
      IF(INMENU(I).EQ. NO) GO TO 1Ø
      K=48Ø-(POSFET(I)-1)*24
      Y = FLOAT(K)
1Ø    RETURN
      END
```

```
                       *         CALL  ERMSG(MSG,LEN,NUM)
0000R                           ENTRY ERMSG
0002                   MSG      EQU   2
0004                   LEN      EQU   4
0006                   NUM      EQU   6
0001                   OUTBUF   EQU   1
0000R                  SAVE     DS    32
0020R  D000            ERMSG    STM   0,SAVE
       0000R
0024R  484F                     LH    LEN,LEN(15)
       0004
0028R  4844                     LH    LEN,0(LEN)
       0000
002CR  0894                     LHR   9,LEN
002ER  4810                     LH    OUTBUF,OUT
       008CR
0032R  482F                     LH    MSG,MSG(15)
       0002
0036R  4882            LOOP     LH    8,0(MSG)
       2000
003AR  4081                     STH   8,0(OUTBUF)
       0000
003ER  C811                     LHI   OUTBUF,2(OUTBUF)
       0002
0042R  C822                     LHI   MSG,2(MSG)
       0002
0046R  CB90                     SHI   9,2
       0002
004AR  4220                     BP    LOOP             LOOP UNTIL MSG IS MOVED
       0036R
004ER  4880                     LH    8,BLANK
       00BAR
0052R  4081                     STH   8,0(OUTBUF)
       0000
0056R  C811                     LHI   OUTBUF,2(OUTBUF)
       0002
005AR  4010                     STH   OUTBUF,DEST
       00BER
005ER  486F                     LH    NUM,NUM(15)      PICK UP NUMBER TO BE PRINTED OUT
       0006
0062R  4806                     LH    0,0(NUM)         PUT IT IN R0 FOR SVC
       0000
0066R  E120                     SVC   2,CONV           CONVERT TO ASCII
       00BCR
006AR  4880                     LH    8,BLANK          GET A COUPLE OF BLANKS
       00BAR
006ER  4081                     STH   8,4(OUTBUF)      STORE IN OUTPUT BUFFER
       0004
0072R  CA40                     AHI   LEN,7            POP CHARACTOR COUNT
       0007
0076R  4040                     STH   LEN,LENGTH       STORE INN PARAMETER LIST
       0090R
007AR  E120                     SVC   2,SEND           PRINT MSG
       008ER
007ER  4300            LOOP2    B     LOOP2            LOOP
       007ER
```

```
0082R D100              LM     0,SAVE
      0040R
0086R 4AF0              AH     15,0(15)
      0000
008AR 030F              BR     15
008CR 0092R    OUT      DC     A(MS)
008ER 0007     SEND     DC     7
0090R 0000     LENGTH   DC     *-*
0092R          MS       DS     40
00BAR 2020     BLANK    DC     X'2020'
00BCR 0006     CONV     DC     6
00BER 0000     DEST     DC     *-*
00C0R                   END
```

```
      SUBROUTINE GRDCEN(I,LEV,NGX,NGY)
      COMMON /CURSTA/ MODE,XCENM,YCENM,OXCEN,OYCEN,XEXTNT,YEXTNT
     1,OXXTNT,OYXTNT
     2,XCURA,YCURA,OXCURA,OYCURA,XCURM,YCURM,OXCURM,OYCURM
      COMMON /DATSTA/ CURLEV(10),AUTOFS(10),GX(10),GY(10),COL(4,10)
      COMMON /DATBAS/ PREFIX(10),NUMLEV(10),INMENU(10),ZMOTHR(4,10),
     1ZMINTH(4,10),D(4,10),NUMX(4,10),NUMY(4,10),IFILE(4,10),
     2 DBINDX(4,10),ITYPE(4,10),ICOLOR(4,10),NUMDB
      COMMON /MAP/MX1,MY1,MX2,MY2
      REAL    MX1,MY1,MX2,MY2
      INTEGER DBINDX
      INTEGER CURLEV,AUTOFS,GX,GY,COL
C   CALCULATE X AXIS GRID POINT CLOSEST TO POINT X FOR CURRENT LEVEL
      NGX = IFIX((ABS(XCENM-MX1) + D(LEV,I)/2.)/D(LEV,I))
C   IF INDEX WITHIN 2 OF MAP EDGE MOVE IT 2 AWAY FROM EDGE
      N=NUMX(LEV,I)-2
      IF(NGX .GT.N) NGX = N
      IF(NGX .LT. 2) NGX = 2
C   CALCULATE Y AXIS GRID POINT CLOSEST TO POINT Y FOR CURRENT LEVEL
      NGY = IFIX((ABS(YCENM-MY1) + D(LEV,I)/2.)/D(LEV,I))
C   IF INDEX WIHTIN 2 OF MAP EDGE MOVE IT 2 AWAY FROM EDGE
      N=NUMY(LEV,I)-2
      IF(NGY .GT. N) NGY= N
      IF(NGY .LT. 2) NGY = 2
      RETURN
      END
```

```
0000R                          ENTRY IMPTAB
0000R                          EXTRN ACORN,SCALE7,ERASE7,TRANS7,CLEAR7,ICURSR,IKEY
0000R                          EXTRN FIND,REFRS7,STATIN,ADITEM
0000R                          EXTRN LSTEN
0000R 0002          IMPTAB     DC    2,C'ACORN   ',0,1,A(ACORN)
      4143
      4F52
      4E20
      2020
      0000
      0001
      0000F
0010R 0002                     DC    2,C'SCALE7  ',0,1,A(SCALE7)
      5343
      414C
      4537
      2020
      0000
      0001
      0000F
0020R 0002                     DC    2,C'ERASE7  ',0,1,A(ERASE7)
      4552
      4153
      4537
      2020
      0000
      0001
      0000F
0030R 0002                     DC    2,C'TRANS7  ',0,1,A(TRANS7)
      5452
      414E
      5337
      2020
      0000
      0001
      0000F
0040R 0002                     DC    2,C'CLEAR7  ',0,1,A(CLEAR7)
      434C
      4541
      5237
      2020
      0000
      0001
      0000F
0050R 0002                     DC    2,C'ICURSR  ',0,1,A(ICURSR)
      4943
      5552
      5352
      2020
      0000
      0001
      0000F
0060R 0002                     DC    2,C'IKEY    ',0,1,A(IKEY)
      494B
      4559
      2020
```

```
      2020
      0000
      0001
      000F
0070R 0002            DC     2,C'FIND    ',0,1,A(FIND)
      4649
      4E44
      2020
      2020
      0000
      0001
      000F
0080R 0002            DC     2,C'REFRS7  ',0,1,A(REFRS7)
      5245
      4652
      5337
      2020
      0000
      0001
      000F
0090R 0002            DC     2,C'**      ',45,1,A(LSTFN)
      2A2A
      2020
      2020
      2020
      002D
      0001
      000F
00A0R 0002            DC     2,C'ADITEM  ',0,1,A(ADITEM)
      4144
      4954
      454D
      2020
      0000
      0001
      000F
00B0R 0002            DC     2,C'STATIN  ',0,1,A(STATIN)
      5354
      4154
      494E
      2020
      0000
      0001
      000F
00C0R 0000            DC     0,0
      0000
00C4R                 END
```

```
0000R                          ENTRY IMPTAB
0000R                          EXTRN ONR,CURSTA
0000R  0002       IMPTAB       DC    2,C'ONR      ',0,0,A(ONR)
       4F4E
       5220
       2020
       2020
       0000
       0000
       0000F
0010R  0002                    DC    2,C'CURSTA   ',0,0,A(CURSTA)
       4355
       5253
       5441
       2020
       0000
       0000
       0000F
0020R  0000                    DC    0,0
       0000
0024R                          END
```

INIT

```
 COMMON /COLORS/ RED,YELLOW,GREEN,BLACK
 COMMON /COMMUN/ SELECT(10),DELETE(10)
 COMMON /CURSTA/ MODE,XCENM,YCENM,OXCEN,OYCEN,XEXTNT,YEXTNT
1,OXXTNT,OYXTNT
2,XCURA,YCURA,OXCURA,OYCURA,XCURM,YCURM,OXCURM,OYCURM
 COMMON /DATBAS/ PREFIX(10),NUMLEV(10),INMENU(10),ZMOTHR(4,10),
1ZMINTH(4,10),D(4,10),NUMX(4,10),NUMY(4,10),IFILE(4,10),
2 DBINDX(4,10),ITYPE(4,10),ICOLOR(4,10),NUMDB
 COMMON /DATSTA/ CURLEV(10),AUTOFS(10),GX(10),GY(10),COL(4,10)
 COMMON /ERASE/ ERSAR(2,40),ICNT,ERSIZE
 COMMON /MACRO/ MACNUM(10),MACEXP(10,4),FETPOS(10),POSFET(10)
1 ,NUMFET
 COMMON /MAP/MX1,MY1,MX2,MY2
 COMMON /MENCON/ MENU,ONXC,ONRXC,OFFXC,OFFRXC,MENNME(2),
1  STATUS(2),SYSTAT(2),SYSRES(2)
2,STATY,RESYC,RLEFT
 COMMON /MNUTIA/ONN,OFF,YES,NO,UP,PASS
1 ,AUTON,AUTOF,STATIC
 COMMON /TRELS/ MAPTRE,MENTRE,WORLD(2),ZINBUT,TRNBUT,SLCTBT,
1 AUFBUT,ZOTBUT,MENBUT,AONBUT,STABUT
2 ,RMAP(2)
 COMMON /FAC/ ZOOMIN,ZOOMOT
 COMMON /FILE/ MFILE
 INTEGER RED,YELLOW,GREEN,BLACK
 INTEGER SELECT,DELETE
 INTEGER DBINDX
 INTEGER CURLEV,AUTOFS,GX,GY,COL
 INTEGER ERSIZE
 INTEGER FETPOS,POSFET
 REAL MX1,MY1,MX2,MY2
 REAL MENNME
 INTEGER AUTON,AUTOF,STATIC
 INTEGER ONN,OFF,YES,NO,UP,PASS
 INTEGER ZINBUT,TRNBUT,SLCTBT ,AUFBUT,ZOTBUT,AONBUT,STABUT
 INTEGER BUFSI(4)
 DIMENSION STAT(21)
 EXTERNAL SAVEA,WRKSP
 EXTERNAL MENUUP,MESLCT,AUTONS,ATOFFS,STATCS
 DATA BUFSI(1),BUFSI(2),BUFSI(3),BUFSI(4)/80,4,1500,3/
 DATA IIOP/-1/
 CALL REDCOM
 CALL ORMBFA(630)
 CALL IMPINT(BUFSI)
 CALL ON(MAPTRE,MENBUT,1,MENUUP)
 CALL ON(MAPTRE,SLCTBT,1,MESLCT)
 CALL ON(MAPTRE,AONBUT,1,AUTONS)
 CALL ON(MAPTRE,AUFBUT,1,ATOFFS)
 CALL ON(MAPTRE,STABUT,1,STATCS)
 STAT(1)=XCENM
 STAT(2)=YCENM
 STAT(3)=XEXTNT
 STAT(4)=YEXTNT
 STAT(5)=XCURA
 STAT(6)=YCURA
```

```
      STAT(7)=XCURM
      STAT(8)=YCURM
      DO 10 J=1,8
      K=J+8
      STAT(K)=STAT(J)
0     CONTINUE
      STAT(17) = WORLD(1)
      STAT(18)=WORLD(2)
      STAT(19) = ZOOMIN
      STAT(20)=ZOOMOT
      STAT(21)=MAPTRE
      CALL SETSAV(WRKSP,SAVEA)
      CALL INTEMP(MFILE,1,SAVEA)
      CALL CHKSAV(SAVEA)
      CALL OPEN(WRKSP,8HMAPMENU  ,SAVEA)
      CALL CHKSAV(SAVEA)
      CALL CLEAR (MENTRE)
      CALL SEND(8HSTATIN   ,63,50,STAT,84,ERROR)
      CALL DISPLY(MAPTRE,ITOP,WORLD,0.,0.,100.,100.,0.,0.,0.,WORLD,0)
      CALL FIND(MAPTRE,WORLD)
      CALL MDISP(IFLAG)
      CALL MP
      END
```

```
0000R                       ENTRY INMVE
                  *          CALL INMVE(BUF(IPOS),INDEX)
0000R             SAVE      DS    16
0010R  D080       INMVE     STM   8,SAVE
       0000R
0014R  488F                 LH    8,2(15)          IBUF ADDRESS
       0002
0018R  489F                 LH    9,4(15)          INDEX ADDRESS
       0004
001CR  C8A0                 LHI   10,8             COUNT OF 4 ENTRIES OF 2 WRDS APIECE
       0008
0020R  48B8       LOOP      LH    11,0(8)          LOAD WORD FROM IBUF
       0000
0024R  40B9                 STH   11,0(9)          STORE IN INDEX
       0000
0028R  C888                 LHI   8,2(8)           POINT TO NEXT BUFFER ENTRY
       0002
002CR  C899                 LHI   9,4(9)           POINT TO NEXT INDEX ENTRY
       0004
0030R  CBA0                 SHI   10,1             DONE 4 ENTRIES YET
       0001
0034R  4230                 BNZ   LOOP             CONTINUE UNTIL DONE
       0020R
0038R  D180                 LM    8,SAVE
       0000R
003CR  4AFF                 AH    15,0(15)
       0000
0040R  030F                 BR    15
0042R                       END
```

```
      SUBROUTINE MDISP(IFORCE)
      COMMON /ERASE/ ERSAR(2,40),ICNT,ERSIZE
      COMMON /MNUTIA/ ON,OFF,YES,NO,UP,PASS
     1 ,AUTON,AUTOF,STATIC
      COMMON /COMMUN/ SELECT(10),DELETE(10)
      COMMON /DATSTA/ CURLEV(10),AUTOFS(10),GX(10),GY(10),COL(4,10)
      COMMON /DATBAS/ PREFIX(10),NUMLEV(10),INMENU(10),ZMOTHR(4,10),
     1ZMINIH(4,10),D(4,10),NUMX(4,10),NUMY(4,10),IFILE(4,10),
     2 DBINDX(4,10),ITYPE(4,10),ICOLOR(4,10),NUMDB
      COMMON /TREES/ MAPTRE,MENTRE,WORLD(2),ZINBUT,TRNBUT,SLCTBT,
     1 AUFBUT,ZOTBUT,MENBUT,AONBUT,STABUT
     2 ,RMAP(2)
      INTEGER ZINBUT,TRNBUT,SLCTBT ,AUFBUT,ZOTBUT,AONBUT,STABUT
      INTEGER CURLEV,AUTOFS,GX,GY,COL
      INTEGER DBINDX
      INTEGER SELECT,DELETE
      INTEGER  ON,OFF,YES,NO,UP,PASS
      INTEGER AUTON,AUTOF,STATIC
      INTEGER ERSIZE
      DIMENSION INDEX(2,4)
      DATA IADDR,LENGTH/1,2/
      DATA ISTART,IEND /1,4/
      DO 20 I= 1,NUMDB
         IF((CURLEV(I) .EQ. 0) .OR. (DELETE(I) .EQ. OFF)) GO TO 10
         CALL CLMERS(I,ISTART,IEND)
      CURLEV(I) = 0
10    DELETE(I) = OFF
20    CONTINUE
      IF(ICNT .EQ. 0) GO TO 30
      CALL ERASE(MAPTRE,ERSAR,ICNT,2)
      ICNT = 0
      IFORCE=1
30    DO 70 I = 1,NUMDB
         IF(SELECT(I) .EQ. OFF) GO TO 70
         LEVEL = SELECT(I)
         CALL GRDCEN(I,LEVEL,GX(I),GY(I))
         CALL SETINX(I,LEVEL,GX(I),GY(I),LEFTOP,INDEX,IBUF,NONE)
         IF (NONE .EQ. 0) GO TO 40
         DO 38 ICOL= 1,4
            COL(ICOL,I) = 0
38       CONTINUE
         GO TO 70
40       DO 60 ICOL= 1,4
            COL(ICOL,I) = LEFTOP + ICOL-1
         IPNT=IBUF
         CALL REDSND(I,LEVEL,INDEX,IPNT,ICOL)
60       CONTINUE
         IFORCE=1
         CURLEV(I) = LEVEL
         SELECT(I) = OFF
         CALL DEALOC
70    CONTINUE
      RETURN
      END
```

```
      SUBROUTINE MENUUP(MSG)
      COMMON /CURSTA/ MODE,XCENM,YCENM,OXCEN,OYCEN,XEXTNT,YEXTNT
     1,OXXTNT,OYXTNT
     2,XCURA,YCURA,OXCURA,OYCURA,XCURM,YCURM,OXCURM,OYCURM
      COMMON /MENCON/ MENU,ONXC,ONRXC,OFFXC,OFFRXC,MENNME(2),
     1  STATUS(2),SYSTAT(2),SYSRES(2)
     2,STATY,RESYC,RLEFT
      COMMON /ANUTIA/ ON,OFF,YES,NO,UP,PASS
     1 ,AUTON,AUTOF,STATIC
      COMMON /DATSTA/ CURLEV(10),AUTOFS(10),GX(10),GY(10),COL(4,10)
      COMMON /DATBAS/ PREFIX(10),NUMLEV(10),INMENU(10),ZMOTHR(4,10),
     1ZMINTH(4,10),D(4,10),NUMX(4,10),NUMY(4,10),TITLE(4,10),
     2 DBINDX(4,10),TIYPE(4,10),ICOLOR(4,10),NUMDB
      COMMON /TREES/ MAPTRE,MENTRE,WORLD(2),ZINBUT,TRNBUT,SLCTBT,
     1 AUFBUT,ZOTBUT,MENBUT,AONBUT,STABUT
     2 ,RMAP(2)
      COMMON /COLORS/ RED,YELLOW,GREEN,BLACK
      REAL MENNME
      INTEGER ZINBUT,TRNBUT,SLCTBT ,AUFBUT,ZOTBUT,AONBUT,STABUT
      INTEGER DBINDX
      INTEGER CURLEV,AUTOFS,GX,GY,COL
      INTEGER AUTON,AUTOF,STATIC
      INTEGER  ON,OFF,YES,NO,UP,PASS
      INTEGER RED,YELLOW,GREEN,BLACK
      DIMENSION MSG(1),MENNM(2)
C IF MENU ALREADY DISPLAYED SKIP STATUS REPORT GENERATION
      IF (MENU .EQ. UP) GO TO 20
      CALL DISPLY(MENTRE,-1,MENNME,0.,0.,100.,100.,0.,0.,0.,MENNME,0)
C GENERATE STATUS REPORT
      CALL OPENI(STATUS,0.,0.,511.,479.,0)
         DO 10 I=1,NUMDB
            IF (INMENU(I) .EQ. NO) GO TO 10
            IF((CURLEV(I) .EQ. 0) .AND. (AUTOFS(I) .EQ. OFF))GO TO 10
            CALL CHARLV(I,ICHAR)
            CALL DBPOS(I,Y)
            CALL NAME(PREFIX(I) ,4HON   ,MENNM)
            CALL CHAR(MENNM,ICHAR,1,ONXC,Y,RED,1,1,0)
10       CONTINUE
C   WRITE MODE AND SELECTION OPTION
         IF (MODE .EQ. AUTON) CALL SETSTA(23HAUTO ON    STATUS REPORT,23)
         IF(MODE.EQ.AUTOF )CALL SETSTA(24HNORMAL      MAKE SELECTION,24)
         IF(MODE.EQ.STATIC) CALL SETSTA(23HSPECIAL   MAKE SELECTION,23)
         CALL CHAR(SYSRES,2H  ,2,RLEFT,RESYC,BLACK,1,1,0)
      CALL DISPLY(MENTRE,MENNME,PASS,0.,0.,511.,479.,0.,0.,0.,STATUS,-1)
      MENU = UP
      IF (MODE .EQ. AUTON) GO TO 15
      GO TO 30
C ENTER MENU AND FORCE DISPLAY CHANGES
15    CALL REFRSH(MENTRE,4,0)
20    CALL CLEAR(MENTRE)
      MENU = OFF
      CALL REFRSH (MENTRE,1,0)
      CALL REFRSH(MAPTRE,4,0)
      IFORCE=0
      CALL MDISP(IFORCE)
      CALL REFRSH(MAPTRE,1,IFORCE)
30    RETURN
      END
```

```
      SUBROUTINE MESLCT(MSG)
      COMMON /MACRO/ MACNUM(1Ø),MACEXP(1Ø,4),FETPOS(1Ø),POSFET(1Ø)
     1 ,NUMFET
      COMMON /DATSTA/ CURLEV(1Ø),AUTOFS(1Ø),GX(1Ø),GY(1Ø),COL(4,1Ø)
      COMMON /DATBAS/ PREFIX(1Ø),NUMLEV(1Ø),INMENU(1Ø),ZMOTHR(4,1Ø),
     1ZMINTH(4,1Ø),D(4,1Ø),NUMX(4,1Ø),NUMY(4,1Ø),IFILE(4,1Ø),
     2 DBINDX(4,1Ø),ITYPE(4,1Ø),ICOLOR(4,1Ø),NUMDB
      COMMON /MNUTIA/ ON,OFF,YES,NO,UP,PASS
     1 ,AUTON,AUTOF,STATIC
      COMMON /MENCON/ MENU,ONXC,ONRXC,OFFXC,OFFRXC,MENNME(2),
     1  STATUS(2),SYSTAT(2),SYSRES(2)
     2,STATY,RESYC,RLEFT
      COMMON /COMMUN/ SELECT(1Ø),DELETE(1Ø)
      COMMON /COLORS/ RED,YELLOW,GREEN,BLACK
      INTEGER RED,YELLOW,GREEN,BLACK
      INTEGER FETPOS,POSFET
      INTEGER CURLEV,AUTOFS,GX,GY,COL
      INTEGER DBINDX
      INTEGER  ON,OFF,YES,NO,UP,PASS
      INTEGER AUTON,AUTOF,STATIC
      INTEGER SELECT,DELETE
      REAL MENNME
      DIMENSION MSG(8)
C IF MENU NOT UP, FUNCTION IS NOOP
      IF (MENU .NE. UP) GO TO 9Ø
      CALL CURPOS(MSG(1),MSG(2),IDB,IACT)
C BE SURE CURSOR SELECTED A FEATURE AND ACTION
      IF (IDB .NE. Ø) GO TO 1Ø
         CALL RESPON(33HCURSOR NOT ALIGNED WITH A FEATURE,33,RED)
      GO TO 9Ø
1Ø    IF(IACT.GE. Ø) GO TO 2Ø
      CALL RESPON (34HCURSOR NOT ALIGNED WITH A FUNCTION,34,RED)
      GO TO 9Ø
C SET LOOP FOR MACRO EXPANSION
2Ø    LOOP = MACNUM(IDB)
      DO 8Ø J = 1,LOOP
         I = MACEXP(IDB,J)
C PROCESS ON FUNCTION
         IF (IACT .EQ. OFF) GO TO 5Ø
C REINSTATE FEATURE IF PREVIOUSLY DELETED
               IF(DELETE(I) .EQ. OFF) GO TO 3Ø
                  DELETE(I) = OFF
                  AUTOFS(I)  = ON
                  IF (INMENU(I) .EQ. NO) GO TO 8Ø
                  CALL WRTCHR(I,OFFXC,4HOFF ,2H  ,15)
                  CALL RESPON(27HFEATURE WILL NOT BE DELETED,27,RED)
                  GO TO 8Ø
C IS FEATURE ALREADY SELECTED
3Ø             IF(AUTOFS(I) .EQ. OFF) GO TO 4Ø
               IF(INMENU(I)  .EQ. YES) CALL RESPON(
     1            24HFEATURE ALREADY SELECTED,24,RED)
                  GO TO 8Ø
C SELECT THE FEATURE
4Ø    CALL CLEVEL(I,LEVEL)
```

```
      SELECT(I) = LEVEL
      AUTOFS(I) = ON
      IF (INMENU(I) .EQ. NO) GO TO 80
      CALL CHARLV(I,ICHAR)
      CALL WRTCHR(I,ONXC,4HON  ,ICHAR,YELLOW)
      CALL RESPON(27HFEATURE SELECTED FOR DISPLY,27,YELLOW)
      GO TO 80
C PROCESS DELETE FUNCTIONS
50    IF (AUTOFS(I) .EQ. ON) GO TO 60
         IF (INMENU(I) .EQ. YES) CALL RESPON(
     1   31HFEATURE NOT CURRENTLY DISPLAYED,31,RED)
         GO TO 80
60    IF (SELECT(I) .EQ. OFF) GO TO 70
         AUTOFS(I) = OFF
         SELECT(I) = OFF
      IF (INMENU(I) .EQ. NO ) GO TO 80
      CALL WRTCHR(I,ONXC,4HON  ,2H  ,15)
      CALL RESPON(29HFEATURE WILL NOT BE DISPLAYED,29,RED)
         GO TO 80
C SELECT FEATURE FOR DELETION
70       IF (CURLEV(I) .NE. 0) DELETE(I) = ON
         AUTOFS(I) = OFF
      IF (INMENU(I) .EQ. NO ) GO TO 80
         CALL WRTCHR(I,OFFXC,4HOFF ,2HX ,YELLOW)
      CALL RESPON(23HFEATURE WILL BE DELETED,23,YELLOW)
80    CONTINUE
90    RETURN
      END
```

```
0000R                    ENTRY  MSEND
0000R                    EXTRN  SEND,ERMSG
                 *        CALL  MSEND(NAME,N63,50,BUF,LENGTH,ERROR)
0000R            SAVE    DS     16
0010R D080       MSEND   STM    8,SAVE
      0000R
0014R 488F               LH     8,2(15)
      0002
0018R 4080               STH    8,NAME
      005ER
001CR 488F               LH     8,4(15)
      0004
0020R 4080               STH    8,PRIO
      0060R
0024R 488F               LH     8,6(15)
      0006
0028R 4080               STH    8,TYPE
      0062R
002CR 488F               LH     8,8(15)
      0008
0030R 4888               LH     8,0(8)
      0000
0034R 4080               STH    8,BUF
      0064R
0038R 488F               LH     8,10(15)
      000A
                 *
                 *        CALCULATE LENGTH OF MESSAGE
                 *
003CR 4888               LH     8,0(8)           LOAD LENGTH FROM INDEX
      0000
0040R CC80               SRHL   8,4              GET RID OF ENTRY BYTE
      0004
0044R CD80               SLHL   8,3              MULT BY 8 BYTES PER POINT
      0003
0048R CA80               AHI    8,24+6           ADD LENGTH OF HEADER AND TRAILER
      001E
004CR 4080               STH    8,LENGTH         STORE AS PARAMETER
      008CR
0050R 488F               LH     8,12(15)
      000C
0054R 4080               STH    8,ERROR
      0068R
0058R 41F0               BAL    15,SEND
      0000F
005CR 000E               DC     14
005ER 0000       NAME    DC     *-*
0060R 0000       PRIO    DC     *-*
0062R 0000       TYPE    DC     *-*
0064R 0000       BUF     DC     *-*
0066R 008CR      LENG    DC     LENGTH
0068R 0000       ERROR   DC     *-*
006AR 4899               LH     9,ERROR
      0068R
006ER 4899               LH     9,0(9)
      0000
```

```
0072R 4330              BZ     RETURN
      0082R
0076R 41F0              BAL    15,ERMSG
      000F
007AR 0008              DC     8
007CR 008ER             DC     A(MSG1)
007ER 0098R             DC     A(TEN)
0080R 0068R             DC     A(ERROR)
0082R D180     RETURN   LM     8,SAVE
      000R
0086R 4AFF              AH     15,0(15)
      0000
008AR 030F              BR     15
008CR 0000     LENGTH   DC     ***
008ER 4D53     MSG1     DC     C'MSEND SEND'
      454E
      4420
      5345
      4E44
0098R 000A     TEN      DC     10
009AR                   END
```

```
      SUBROUTINE MTRANS(IFORCE)
      COMMON /MNUTIA/ ON,OFF,YES,NO,UP,PASS
     1 ,AUTON,AUTOF,STATIC
      COMMON /COMMUN/ SELECT(10),DELETE(10)
      COMMON /DATSTA/ CURLEV(10),AUTOFS(10),GX(10),GY(10),COL(4,10)
      COMMON /DATBAS/ PREFIX(10),NUMLEV(10),INMENU(10),ZMOTHR(4,10),
     1ZMINTH(4,10),D(4,10),NUMX(4,10),NUMY(4,10),IFILE(4,10),
     2 DBINDX(4,10),ITYPE(4,10),ICOLOR(4,10),NUMDB
      INTEGER DBINDX
      INTEGER CURLEV,AUTOFS,GX,GY,COL
      INTEGER AUTON,AUTOF,STATIC
      INTEGER  ON,OFF,YES,NO,UP,PASS
      INTEGER SELECT,DELETE
      DO 20 I=1,NUMDB
         IF(CURLEV(I) .EQ. 0) GO TO 20
         CALL GRDCEN(I,CURLEV(I),NGX,NGY)
         IF (NGY .EQ. GY(I)) GO TO 10
            SELECT(I) =CURLEV(I)
            DELETE(I) = ON
            GO TO 20
10       IF(NGX .EQ. GX(I)) GO TO 20
            CALL RPCOL(I,NGX,NGY,IFORCE)
            GX(I) =NGX
20    CONTINUE
      CALL MDISP(IFORCE)
      RETURN
      END
```

```
0000R                       ENTRY  NAME
0000              NUMAR     EQU    0
0002              FSTNM     EQU    2
0004              SECNM     EQU    4
0006              NAMES     EQU    6
0000R             SAVE      DS     32
0020R  D000       NAME      STM    0,SAVE
       0000R
0024R  486F                 LH     NAMES,NAMES(15)
       0006
0028R  482F                 LH     FSTNM,FSTNM(15)
       0002
002CR  484F                 LH     SECNM,SECNM(15)
       0004
0030R  4812                 LH     1,0(FSTNM)
       0000
0034R  4016                 STH    1,0(NAMES)
       0000
0038R  4812                 LH     1,2(FSTNM)
       0002
003CR  4016                 STH    1,2(NAMES)
       0002
0040R  0711                 XHR    1,1
0042R  4016                 STH    1,4(NAMES)
       0004
0046R  4814                 LH     1,0(SECNM)
       0000
004AR  4016                 STH    1,6(NAMES)
       0006
004ER  D100                 LM     0,SAVE
       0000R
0052R  4AFF                 AH     15,NUMAR(15)
       0000
0056R  030F                 BR     15
0058R                       END
```

```
      SUBROUTINE NEWCEN(FAC,OXCENM,OYCENM,XCENM,YCENM)
      COMMON /STATUS/ CURSTA(16)
      INTEGER XCENN,YCENN,XXTNT,YXTNT,XCURA,YCURA,XCURM,YCURM
      INTEGER OXCENM,OYCENM,OXXTNT,OYXTNT,OXCURA,OYCURA,OXCURM,OYCURM
      DATA XCENN,YCENN,XXTNT,YXTNT,XCURA,YCURA,XCURM,YCURM,
     1 OXCENM,OYCENM,OXXTNT,OYXTNT,OXCURA,OYCURA,OXCURM,OYCURM
     2  /1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16/
      XCENM= CURSTA(XCURM)-(CURSTA(XCURM)-OXCENM)*FAC
      YCENM=CURSTA(YCURM)-(CURSTA(YCURM)-OYCENM)*FAC
      RETURN
      END
```

```
ØØØØR                    ENTRY REDCOM
ØØØØR           SAVE     DS    32
Ø2C4            COMBOT   EQU   X'2C4'
Ø2C2            CTOP     EQU   X'2C2'
ØØ2ØR DØØØ      REDCOM   STM   Ø,SAVE
      ØØ3ØR
ØØ24R E11Ø               SVC   1,RWND
      ØØ86R
ØØ28R 483Ø               LH    3,RWND+2
      ØØ88R
ØØ2CR 433Ø               BZ    GO
      ØØ3CR
ØØ3ØR E12Ø               SVC   2,UNPCK
      ØØ92R
ØØ34R E12Ø               SVC   2,SEND
      ØØ96R
ØØ38R E13Ø               SVC   3,Ø
      ØØØØ
ØØ3CR E11Ø      GO       SVC   1,READ1
      ØØ7ER
ØØ4ØR 48ØØ               LH    Ø,READ1+2
      ØØ8ØR
ØØ44R 423Ø               BNZ   ERR
      ØØ64R
ØØ48R 481Ø               LH    1,LEN
      ØØ7AR
ØØ4CR 441Ø               STH   1,START
      ØØ8ER
ØØ5ØR 481Ø               LH    1,CTOP
      Ø2C2
ØØ54R 401Ø               STH   1,END
      ØØ9ØR
ØØ58R E11Ø               SVC   1,READ
      ØØ8AR
ØØ5CR 48ØØ               LH    Ø,STAT
      ØØ8CR
ØØ6ØR 433Ø               BZ    RETURN
      ØØ7ØR
ØØ64R E12Ø      ERR      SVC   2,UNPCK
      ØØ92R
ØØ68R E12Ø               SVC   2,SEND
      ØØ96R
ØØ6CR E13Ø               SVC   3,Ø
      ØØØØ
ØØ7ØR D1ØØ      RETURN   LM    Ø,SAVE
      ØØØØR
ØØ74R 4AFF               AH    15,Ø(15)
      ØØØØ
ØØ78R Ø3ØF               BR    15
ØØ7AR           LEN      DS    4
ØØ7ER 58Ø6      READ1    DC    X'58Ø6'
ØØ8ØR ØØØØ               DC    *-*
ØØ82R ØØ7AR              DC    A(LEN)
ØØ84R ØØ7DR              DC    A(LEN)+3
ØØ86R C3Ø6      RWND     DC    X'C3Ø6'
ØØ88R ØØØØ               DC    *-*
```

```
ØØ8AR 58Ø6     READ     DC     X'58Ø6'
ØØ8CR ØØØØ     STAT     DC     *-*
ØØ8ER ØØØØ     START    DC     *-*
ØØ9ØR ØØØØ     END      DC     *-*
ØØ92R ØØØ6     UNPCK    DC     6
ØØ94R ØØ9AR             DC     A(MSG)
ØØ96R ØØØ7     SEND     DC     7
ØØ98R ØØØ4              DC     4
ØØ9AR          MSG      DS     4
ØØ9ER                   END
```

```
      SUBROUTINE REDSND(I,LEVEL,INDEX,IPNT,ICOL)
      COMMON /DATSTA/ CURLEV(10),AUTOFS(10),GX(10),GY(10),COL(4,10)
      COMMON /DATBAS/ PREFIX(10),NUMLEV(10),INMENU(10),ZMOTHR(4,10),
     1ZMINTH(4,10),D(4,10),NUMX(4,10),NUMY(4,10),IFILE(4,10),
     2 DBINDX(4,10),ITYPE(4,10),ICOLOR(4,10),NUMDB
      COMMON /TREES/ MAPTRE,MENTRE,WORLD(2),ZINBUT,TRNBUT,SLCTBT,
     1 AUFBUT,ZOTBUT,MENBUT,AONBUT,STABUT
     2 ,RMAP(2)
      INTEGER ZINBUT,TRNBUT,SLCTBT ,AUFBUT,ZOTBUT,AONBUT,STABUT
      INTEGER CURLEV,AUTOFS,GX,GY,COL
      INTEGER DBINDX
      INTEGER RFRSH
      DIMENSION INDEX(2,4)
      DIMENSION NAMES(2)
      DATA RFRSH /-1/
      DATA IADDR,LENGTH /1,2/
      CALL RETREV(IFILE(LEVEL,I),INDEX(IADDR,ICOL),IPNT,NONE)
      IF(NONE .EQ. 0) GO TO 10
      COL(ICOL,I) = 0
      GO TO 20
10    IBUF=IPNT
      CALL SETBF (MAPTRE,RMAP ,RFRSH,IPNT)
      CALL NAME(PREFIX(I),COL(ICOL,I),NAMES)
      CALL SETITM(ITYPE(LEVEL,I),INDEX(LENGTH,ICOL),ICOLOR(LEVEL,I),
     1  NAMES,IPNT)
      CALL MSEND(8HADITEM  , 63,50,IBUF,INDEX(LENGTH,ICOL),ERROR)
20    RETURN
      END
```

```
      SUBROUTINE RESPON(ICHAR,LEN,ICOLOR)
      COMMON /MNUTIA/ ON,OFF,YES,NO,UP,PASS
     1 ,AUTON,AUTOF,STATIC
      COMMON /MENCON/ MENU,ONXC,ONRXC,OFFXC,OFFRXC,MENNME(2),
     1  STATUS(2),SYSTAT(2),SYSRES(2)
     2,STATY,RESYC,RLEFT
      COMMON /TREES/ MAPTRE,MENTRE,WORLD(2),ZINBUT,TRNBUT,SLCTBT,
     1 AUFBUT,ZOTBUT,MENBUT,AONBUT,STABUT
     2 ,RMAP(2)
      INTEGER ZINBUT,TRNBUT,SLCTBT ,AUFBUT,ZOTBUT,AONBUT,STABUT
      INTEGER AUTON,AUTOF,STATIC
      INTEGER  ON,OFF,YES,NO,UP,PASS
      REAL MENNME
      CALL ERASE(MENTRE,SYSRES,1,-1)
      CALL OPENT(SYSRES,0.,0.,511.,479.,0)
      CALL CHAR(8HSYSRESCH,ICHAR,LEN,RLEFT,RESYC,ICOLOR,1,1,0)
      CALL DISPLY(MENTRE,STATUS,PASS,0.,0.,511.,479.,0.,0.,0.,SYSRES,-1)
      RETURN
      END
```

```
0000R                 ENTRY RETREV
              *        CALL RETREV(FILE,INDEX(1,COL),IPNT,NONE)
0000R                 EXTRN DRUM,ERMSG
0002          FILE    EQU   2
0004          INDEX   EQU   4
0006          IPNT    EQU   6
0008          NONE    EQU   8
0000R         SAVE    DS    32
0020R D000    RETREV  STM   0,SAVE
      0000R
0024R 482F            LH    FILE,FILE(15)
      0002
0028R 484F            LH    INDEX,INDEX(15)
      0004
002CR 486F            LH    IPNT,IPNT(15)
      0006
0030R 488F            LH    NONE,NONE(15)
      0008
0034R 4020            STH   FILE,DFILE         STORE ADDR FILE NUMBER IN PARAM LIST
      0076R
0038R 4816            LH    1,0(IPNT)          PIC UP ADDRESS OF ALOCATED SPACE
      0000
003CR 4010            STH   1,BUFFER           PUT IT IN PARAMTER LIST
      0078R
0040R 4814            LH    1,0(INDEX)         GET BLOCK ADDRES FROM INDEX
      0000
0044R 4010            STH   1,BLK              PUT IT IN PARAMETER LIST
      00DAR
0048R 4814            LH    1,4(INDEX)         GET NUMBER OF ENTRIES FROM PARAMETER LIST
      0004
004CR CC10            SRHL  1,4                SHIFT OUT BYTE ADDRESS
      0004
0050R 4230            BNZ   RETR               CONTINUE IF NOT EMPTY COLUMN
      005CR
0054R 48A0            LH    10,THREE           LOAD EMPTY COL RETURN CODE
      00DCR
0058R 4300            B     RETURN
      00CAR
005CR 4824    RETR    LH    2,4(INDEX)         GET ENTRY ADDRES IN BLOCK
      0004
0060R 4420            NH    2,MASK             GET RID OF POINT COUNT
      00E2R
0064R 0A12            AHR   1,2                ADD TO POINT COUNT TO BE READ
0066R CD10            SLHL  1,1                MULT BY 2 NUMBERS PER ENTRY
      0001
006AR 4010            STH   1,NUM              STORE NUMBER OF ENTRIES IN PARAMETER BLOCK
      00D8R
006ER 41F0            BAL   15,DRUM            READ THE COL FROM DRUM
      0000F
0072R 000E            DC    14
0074R 00DCR           DC    A(THREE)
0076R 0000    DFILE   DC    *-*
0078R 0000    BUFFER  DC    *-*
007AR 00D8R           DC    A(NUM)
007CR 00DAR           DC    A(BLK)
007ER 00DER           DC    A(ERROR)
```

```
ØØBØR 48CØ              LH     12,ERROR
      ØØDER
ØØB4R 433Ø              BZ     CONT
      ØØ94R
ØØB8R 41FØ              BAL    15,ERMSG
      ØØØØF
ØØBCR ØØØ8              DC     8
ØØ8ER ØØE4R             DC     A(MSG1)
ØØ9ØR ØØFØR             DC     A(ELEV)
ØØ92R ØØDER             DC     A(ERROR)
ØØ94R 4814     CONT     LH     1,4(INDEX)      GET COL START LOCATION IN BLOCK READ
      ØØØ4
ØØ98R 441Ø              NH     1,MASK          AND OUT LENGTH
      ØØE2R
ØØ9CR CD1Ø              SLHL   1,3             MULT BY 8 BYTES PER ENTRY
      ØØØ3
ØØAØR 4A1Ø              AH     1,BUFFER        ADD COL START TO BEGINING OF BUFFER
      ØØ78R
ØØA4R CB1Ø              SHI    1,24            ALLOW ROOM FOR 24 BYTE HEADER
      ØØ18
ØØA8R 4Ø16              STH    1,Ø(IPNT)       STORE ADDRESS IN RETURN PARAMETER
      ØØØØ
               *
               *          PUT CLOSE AT END OF BUFFER
               *
ØØACR 483Ø              LH     3,NUM           GET NUMBER OF FORTRAN NUMBERS
      ØØD8R
ØØBØR CD3Ø              SLHL   3,2             MULT BY 4 BYTES PER NUMBER
      ØØØ2
ØØB4R 4A3Ø              AH     3,BUFFER        POINT TO END OF BUFFER
      ØØ78R
ØØB8R C8ØØ              LHI    Ø,-1            GET END OF BUFFER DELEMETER
      FFFF
ØØBCR 4ØØ3              STH    Ø,Ø(3)
      ØØØØ
ØØCØR Ø7AA              XHR    1Ø,1Ø           OTHER DELEMTETERS
ØØC2R 4ØA3              STH    1Ø,2(3)         STORE IN BUFFER
      ØØØ2
ØØC6R 4ØA3              STH    1Ø,4(3)
      ØØØ4
ØØCAR 4ØAØ     RETURN   STH    1Ø,Ø(NONE)      SET RETURN CODE
      ØØØØ
ØØCER D1ØØ              LM     Ø,SAVE
      ØØØØR
ØØD2R 4AFØ              AH     15,Ø(15)
      ØØØØ
ØØD6R Ø3ØF              BR     15
ØØD8R ØØØØ     NUM      DC     *-*
ØØDAR ØØØØ     BLK      DC     *-*
ØØDCR ØØØ3     THREE    DC     3
ØØDER ØØØØ     ERROR    DC     *-*
ØØEØR ØØØØ              DC     *-*             2ND HALFWORD OF ERROR
ØØE2R ØØØF     MASK     DC     X'ØØØF'
ØØE4R 5245     MSG1     DC     C'RETREV DRUM'
      5452
      4556
      2Ø44
      5255
      4D2Ø
ØØFØR ØØØB     ELEV     DC     11
ØØF2R                   END
```

```
      SUBROUTINE RINDEX(I,LFTTOP,LEV,INDEX)
      COMMON /DATSTA/ CURLEV(10),AUTOFS(10),GX(10),GY(10),COL(4,10)
      COMMON /DATBAS/ PREFIX(10),NUMLEV(10),INMENU(10),ZMOTHR(4,10),
     1ZMINTH(4,10),D(4,10),NUMX(4,10),NUMY(4,10),IFILE(4,10),
     2 DBINDX(4,10),ITYPE(4,10),ICOLOR(4,10),NUMDB
      INTEGER CURLEV,AUTOFS,GX,GY,COL
      INTEGER DBINDX
      INTEGER ERROR
      DIMENSION BUF(36),INDEX(2,4)
      IBYTE =(LFTTOP-1) *4
      IBLK=IBYTE/128
      CALL DRUM(3,DBINDX(LEV,I),BUF,36,IBLK,ERROR)
      IF(ERROR .NE. 0) CALL ERMSG(11HRINDEX DRUM,11,ERROR)
      IPOS=IBYTE -IBLK*128
      IPOS=IPOS/4+1
      CALL INMVE(BUF(IPOS),INDEX)
      RETURN
      END
```

```
      SUBROUTINE RPCOL(I,NGX,NGY,IFORCE)
      COMMON /ERASE/ ERSAR(2,40),ICNT,ERSIZE
      COMMON /DATSTA/ CURLEV(10),AUTOFS(10),GX(10),GY(10),COL(4,10)
      COMMON /DATBAS/ PREFIX(10),NUMLEV(10),INMENU(10),ZMOTHR(4,10),
     1ZMINTH(4,10),D(4,10),NUMX(4,10),NUMY(4,10),IFILE(4,10),
     2 DBINDX(4,10),ITYPE(4,10),ICOLOR(4,10),NUMDB
      COMMON /TREES/ MAPTRE,MENTRE,WORLD(2),ZINBUT,TRNBUT,SLCTBT,
     1 AUFBUT,ZOTBUT,MENBUT,AONBUT,STABUT
     2 ,RMAP(2)
      INTEGER ZINBUT,TRNBUT,SLCTBT ,AUFBUT,ZOTBUT,AONBUT,STABUT
      INTEGER DBINDX
      INTEGER CURLEV,AUTOFS,GX,GY,COL
      INTEGER ERSIZE
      DIMENSION ITEMP(4),INDEX(2,4)
      LIM =IABS(NGX-GX(I))
      IF(NGX .GT. GX(I)) GO TO 10
         IEND = 4
         ISTART = 3
         IF(LIM .EQ. 1) ISTART = 4
         IROT=4-LIM
         GO TO 20
10       ISTART = 1
         IEND = 2
         IF(LIM .EQ. 1) IEND = 1
         IROT=LIM
20    CALL CLMERS(I,ISTART,IEND)
      CALL ERASE(MAPTRE,ERSAR,ICNT,2)
      IFORCE=1
      ICNT=0
      DO 50 J=1,IROT
         DO 30 ICOL=1,4
            K=ICOL+1
            IF(ICOL .EQ. 4) K =1
            ITEMP(ICOL) = COL(K,I)
30       CONTINUE
         DO 40 ICOL=1,4
            COL(ICOL,I) = ITEMP(ICOL)
40       CONTINUE
50    CONTINUE
      LEVEL = CURLEV(I)
      CALL SETINX(I,LEVEL,NGX,NGY,LEFTOP,INDEX,IBUF,NONE)
      IF (NONE .NE. 0) GO TO 70
         DO 60 ICOL=1,4
             IF(COL(ICOL,I) .NE. 0) GO TO 60
            COL(ICOL,I) =LEFTOP + ICOL-1
         IPNT=IBUF
            CALL REDSND(I,LEVEL,INDEX,IPNT,ICOL)
60       CONTINUE
      CALL DEALOC
70    RETURN
      END
```

```
                     *        CALL SETBF (TREE,WHERE,REFRSH,IPNT)
0000R                         ENTRY SETBF
0000R                SAVE     DS    16
0000                 RET      EQU   0
0002                 TREE     EQU   2
0004                 WHERE    EQU   4
0006                 REFRSH   EQU   6
0008                 IPNT     EQU   8
0010R D080           SETBF    STM   8,SAVE
      0000R
0014R 48CF                    LH    12,IPNT(15)        PICK UP POINTER ADDRESS
      0008
0018R 488C                    LH    IPNT,0(12)         PICK UP POINTER
      0000
001CR 489F                    LH    9,TREE(15)         GET ADDRESS OF TREE
      0002
0020R 4899                    LH    9,0(9)             GET TREE NUMBER
      0000
0024R 4098                    STH   9,0(IPNT)          PUT TREE IN BUFFER
      0000
0028R C888                    LHI   IPNT,2(IPNT)       POP BUFFER POINTER
      0002
002CR 489F                    LH    9,WHERE(15)        GET ADDRESS OF WHERE NAME
      0004
0030R C8B0                    LHI   11,4               SET LOOP COUNTER
      0004
0034R 48A9           LOOP     LH    10,0(9)            GET 1ST WORD OF WHERE NAME
      0000
0038R 40A8                    STH   10,0(IPNT)         STORE IT IN BUFFER
      0000
003CR C888                    LHI   IPNT,2(IPNT)       POP BUFFER POINTER
      0002
0040R C899                    LHI   9,2(9)             POP NAME POINTER
      0002
0044R CBB0                    SHI   11,1               DONE LOOPING
      0001
0048R 4230                    BNZ   LOOP               NO LOOP AGAIN
      0034R
004CR 489F                    LH    9,REFRSH(15)       GET ADDRES OF REFRESH TYPE
      0006
0050R 4899                    LH    9,0(9)             GET REFRESH TYPE
      0000
0054R 4098                    STH   9,0(IPNT)          STORE IN BUFFER
      0000
0058R C888                    LHI   IPNT,2(IPNT)       POP BUFFER POINTER
      0002
005CR 408C                    STH   IPNT,0(12)         SET BUFFER POINTER IN RETURN PARAMETERS
      0000
0060R D180                    LM    8,SAVE
      0000R
0064R 4AFF                    AH    15,RET(15)
      0000
0068R 030F                    BR    15
006AR                         END
```

```
SUBROUTINE SETINX(I,LEVEL,NGX,NGY,LFTTOP,INDEX,IPNT,NONE)
CALL TOPLFT(I,LEVEL,NGX,NGY,LFTTOP)
CALL RINDEX(I,LFTTOP,LEVEL,INDEX)
CALL ALLOC(INDEX,IPNT,NONE)
RETURN
END
```

```
0000R                   ENTRY SETITM
                *        CALL SETITM(ITYPE,N,COLOR,NAME,IPNT)
0000            RET     EQU   0
0002            ITYPE   EQU   2
0004            N       EQU   4
0006            COLOR   EQU   6
0008            NAME    EQU   8
000A            IPNT    EQU   10
0000R           SAVE    DS    32
0020R D000      SETITM  STM   0,SAVE
      0000R
0024R 481F              LH    1,IPNT(15)
      003A
0028R 48A1              LH    IPNT,0(1)        PICK UP BUFFER POINTER
      0000
002CR 482F              LH    ITYPE,ITYPE(15)
      0002
0030R 4832              LH    3,0(ITYPE)       LOAD TYPE
      0000
0034R CD30              SLHL  3,12             PUT TYPE IN HIGH ORDER BITS
      000C
0038R 484F              LH    N,N(15)          GET ADDRESS OF NUMBER OF POINTS
      0004
003CR 4844              LH    N,0(N)           GET NUMBER OF POINTS
      0000
0040R CC40              SRHL  N,4              GET RID OF BYTE ADDRESS
      0004
0044R 0634              OHR   3,N              MAKE ITEM HEADER OF TYPE AND POINT COUNT
0046R 403A              STH   3,0(IPNT)        PUT IT IN BUFFER
      0000
004AR C8AA              LHI   IPNT,2(IPNT)     POP BUFFER POINTER
      0002
004ER 486F              LH    COLOR,COLOR(15)
      0006
0052R 4866              LH    COLOR,0(COLOR)
      0000
0056R 406A              STH   COLOR,0(IPNT)    PUT COLOR INTO BUFFER
      0000
005AR C8AA              LHI   IPNT,2(IPNT)     POP BUFFER POINTER
      0002
005ER C870              LHI   7,4              SET LOOP COUNTER
      0004
0062R 488F              LH    NAME,NAME(15)    GET ADDRES OF NAME
      0008
0066R 4858      LOOP    LH    5,0(NAME)        PICK UP NAME
      0000
006AR 405A              STH   5,0(IPNT)        STORE IN BUFFER
      0000
006ER C8AA              LHI   IPNT,2(IPNT)     POP BUFFER POINTER
      0002
0072R C888              LHI   NAME,2(NAME)     POP NAME POINTER
      0002
0076R CB70              SHI   7,1              DONE LOOPING
      0001
007AR 4233              BNZ   LOOP             GO LOOP IF NOT DONE
      0066R
```

```
007ER 44A1             STH   IPNT,0(1)        SAVE POINTER IN RETURN PARAMETER
      0000
0082R D100             LM    0,SAVE
      0000R
0086R 4AFF             AH    15,RET(15)
      0000
008AR 030F             BR    15
008CR                  END
```

```
      SUBROUTINE SETMSG(MSG)
      COMMON /STATUS/ CURSTA(16)
      DIMENSION MSG(1)
      INTEGER XCENM,YCENM,XXTNT,YXTNT,XCURA,YCURA,XCURM,YCURM
      INTEGER OXCENM,OYCENM,OXXTNT,OYXTNT,OXCURA,OYCURA,OXCURM,OYCURM
      DATA IX,IY/1,2/
      DATA XCENM,YCENM,XXTNT,YXTNT,XCURA,YCURA,XCURM,YCURM,
     1 OXCENM,OYCENM,OXXTNT,OYXTNT,OXCURA,OYCURA,OXCURM,OYCURM
     2  /1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16/
      DO 10 J= 1,8
         K=J+8
         CURSTA(K) = CURSTA(J)
10    CONTINUE
      CURSTA(XCURA) = FLOAT(MSG(IX))
      CURSTA(YCURA) = FLOAT(MSG(IY))
      CALL CRTOMP(CURSTA(XCURA),CURSTA(YCURA),
     1   CURSTA(XCURM),CURSTA(YCURM))
      RETURN
      END
```

```
      SUBROUTINE SETSTA(ICHAR,LEN)
      COMMON /COLORS/ RED,YELLOW,GREEN,BLACK
      COMMON /MENCON/ MENU,ONXC,ONRXC,OFFXC,OFFRXC,MENNME(2),
     1  STATUS(2),SYSTAT(2),SYSRES(2)
     2,STATY,RESYC,RLEFT
      INTEGER RED,YELLOW,GREEN,BLACK
      REAL MENNME
      CALL CHAR(SYSTAT,ICHAR,LEN,RLEFT,STATY,GREEN,1,1,0)
      RETURN
      END
```

```
      SUBROUTINE STATCS(MSG)
      COMMON /MNUTIA/ ON,OFF,YES,NO,UP,PASS
     1 ,AUTON,AUTOF,STATIC
      COMMON /CURSTA/ MODE,XCENM,YCENM,OXCEN,OYCEN,XEXTNT,YEXTNT
     1,OXXTNT,OYXTNT
     2,XCURA,YCURA,OXCURA,OYCURA,XCURM,YCURM,OXCURM,OYCURM
      INTEGER  ON,OFF,YES,NO,UP,PASS
      INTEGER AUTON,AUTOF,STATIC
      MODE = STATIC
      RETURN
      END
```

```
      SUBROUTINE STATIN(NAME,TYPE,LENGTH,STAT)
      COMMON /DATA / WORLD(2),ZOOMIN,ZOOMOT,MAPTRE
      COMMON /STATUS/ CURSTA(16)
      REAL NAME
      INTEGER TYPE
      DIMENSION STAT(1)
      DATA I1,I2,I3,I4,I5/17,18,19,20,21/
      DO 10 J=1,16
      CURSTA(J) =STAT(J)
10    CONTINUE
      WORLD(1)=STAT(I1)
      WORLD(2)=STAT(I2)
      ZOOMIN=STAT(I3)
      ZOOMOT=STAT(I4)
      MAPTRE=STAT(I5)
      RETURN
      END
```

```
      SUBROUTINE TOPLFT(I,LEVEL,NGX,NGY,LFTTOP)
      COMMON /DATSTA/ CURLEV(10),AUTOFS(10),GX(10),GY(10),COL(4,10)
      COMMON /DATBAS/ PREFIX(10),NUMLEV(10),INMENU(10),ZMOTHR(4,10),
     1ZMINTH(4,10),D(4,10),NUMX(4,10),NUMY(4,10),IFILE(4,10),
     2 DBINDX(4,10),ITYPE(4,10),ICOLOR(4,10),NUMDB
      INTEGER CURLEV,AUTOFS,GX,GY,COL
      INTEGER DBINDX
      LFTTOP = ((NGY+1)  * NUMX(LEVEL,I)) + (NGX-1)
      RETURN
      END
```

```
      SUBROUTINE TRANTP(MSG)
      COMMON /DATA / WORLD(2),ZOOMIN,ZOOMOT,MAPTRE
      COMMON /STATUS/ CURSTA(16)
      DIMENSION MSG(1)
      INTEGER ERROR
      INTEGER XCENM,YCENM,XXTNT,YXTNT,XCURA,YCURA,XCURM,YCURM
      INTEGER OXCENM,OYCENM,OXXTNT,OYXTNT,OXCURA,OYCURA,OXCURM,OYCURM
      DATA XCENM,YCENM,XXTNT,YXTNT,XCURA,YCURA,XCURM,YCURM,
     1 OXCENM,OYCENM,OXXTNT,OYXTNT,OXCURA,OYCURA,OXCURM,OYCURM
     2  /1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16/
      CALL SETMSG(MSG)
      DISTX = 256.-CURSTA(XCURA)
      DISTY = 240. -CURSTA(YCURA)
      CALL TRANS(MAPTRE,  WORLD   ,DISTX,DISTY,0,1)
      CURSTA(XCENM) = CURSTA(XCURM)
      CURSTA(YCENM) = CURSTA(YCURM)
      CALL SEND(8HCURSTA    ,63,50,CURSTA,64,ERROR)
      IF(ERROR .NE. 0)CALL ERMSG (11HTRANTP SEND,11,ERROR)
      RETURN
      END
```

```
      SUBROUTINE WRTCHR(I,X,SECNME,ICHAR,ICOLR)
      COMMON /MNUTIA/ ON,OFF,YES,NO,UP,PASS
     1 ,AUTON,AUTOF,STATIC
      COMMON /TREES/ MAPTRE,MENTRE,WORLD(2),ZINBUT,TRNBUT,SLCTBT,
     1 AUFBUT,ZOTBUT,MENBUT,AONBUT,STABUT
     2 ,RMAP(2)
      COMMON /DATSTA/ CURLEV(10),AUTOFS(10),GX(10),GY(10),COL(4,10)
      COMMON /DATBAS/ PREFIX(10),NUMLEV(10),INMENU(10),ZMOTHR(4,10),
     1ZMINTH(4,10),D(4,10),NUMX(4,10),NUMY(4,10),IFILE(4,10),
     2 DBINDX(4,10),ITYPE(4,10),ICOLOR(4,10),NUMDB
      COMMON /MENCON/ MENU,ONXC,ONRXC,OFFXC,OFFRXC,MENNME(2),
     1  STATUS(2),SYSTAT(2),SYSRES(2)
     2,STATY,RESYC,RLEFT
      INTEGER  ON,OFF,YES,NO,UP,PASS
      INTEGER AUTON,AUTOF,STATIC
      INTEGER ZINBUT,TRNBUT,SLCTBT ,AUFBUT,ZOTBUT,AONBUT,STABUT
      INTEGER CURLEV,AUTOFS,GX,GY,COL
      INTEGER DBINDX
      REAL MENNME
      DIMENSION CNAME(2)
      IF (INMENU(I) .EQ. NO) GO TO 10
      CALL DBPOS(I,Y)
      CALL NAME(PREFIX(I),SECNME,CNAME)
      CALL OPENT(CNAME,0.,0.,511.,479.,0)
      CALL CHAR(CNAME,ICHAR,2,X,Y,ICOLR ,1,1,0)
      CALL DISPLY(MENTRE,STATUS,PASS,0.,0.,511.,479.,0.,0.,0.,CNAME,-1)
10    RETURN
      END
```

```
SUBROUTINE ZMINTP(MSG)
COMMON /DATA / WORLD(2),ZOOMIN,ZOOMOT,MAPTRE
DIMENSION MSG(1)
CALL ZOMTOP(MSG,ZOOMIN)
RETURN
END
```

```
SUBROUTINE ZMOUTP(MSG)
COMMON /DATA / WORLD(2),ZOOMIN,ZOOMOT,MAPTRE
DIMENSION MSG(1)
CALL ZOMTOP(MSG,ZOOMOT)
RETURN
END
```

```
      SUBROUTINE ZMTRNS
      COMMON /CURSTA/ MODE,XCENM,YCENM,OXCEN,OYCEN,XEXTNT,YEXTNT
     1,OXXTNT,OYXTNT
     2,XCURA,YCURA,OXCURA,OYCURA,XCURM,YCURM,OXCURM,OYCURM
      COMMON /MNUTIA/ ON,OFF,YES,NO,UP,PASS
     1 ,AUTON,AUTOF,STATIC
      COMMON /TREES/ MAPTRE,MENTRE,WORLD(2),ZINBUT,TRNBUT,SLCTBT,
     1 AUFBUT,ZOTBUT,MENBUT,AONBUT,STABUT
     2 ,RMAP(2)
      INTEGER ZINBUT,TRNBUT,SLCTBT ,AUFBUT,ZOTBUT,AONBUT,STABUT
      INTEGER  ON,OFF,YES,NO,UP,PASS
      INTEGER AUTON,AUTOF,STATIC
      IFORCE=0
      CALL REFRSH(MAPTRE,0,0)
      IF(XEXTNT .EQ. OXXTNT) GO TO 10
      IF(MODE.EQ.AUTON) CALL AUTONZ
      IF(MODE .EQ. AUTOF ) CALL AUTOFZ
      IF(MODE.NE. STATIC) CALL MDISP(IFORCE)
10    CALL MTRANS(IFORCE)
      CALL REFRSH(MAPTRE,1,IFORCE)
      RETURN
      END
```

```
      SUBROUTINE ZOMTOP(MSG,FAC)
      COMMON /STATUS/ CURSTA(16)
      COMMON /DATA / WORLD(2),ZOOMIN,ZOOMOT,MAPTRE
      INTEGER XCENM,YCENM,XXTNT,YXTNT,XCURA,YCURA,XCURM,YCURM
      INTEGER OXCENM,OYCENM,OXXTNT,OYXTNT,OXCURA,OYCURA,OXCURM,OYCURM
      INTEGER ERROR
      DIMENSION MSG(1)
      DATA XCENM,YCENM,XXTNT,YXTNT,XCURA,YCURA,XCURM,YCURM,
     1 OXCENM,OYCENM,OXXTNT,OYXTNT,OXCURA,OYCURA,OXCURM,OYCURM
     2  /1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16/
      CALL SETMSG(MSG)
      FACTOR = 1./FAC
      CALL SCALE(MAPTRE,FACTOR,CURSTA(XCURA),CURSTA(YCURA),  WORLD
     10,1)
      CURSTA(XXTNT) =CURSTA(OXXTNT) *FAC
      CURSTA(YXTNT ) = CURSTA(OYXTNT) *FAC
      CALL NEWCEN(FAC,CURSTA(OXCENM),CURSTA(OYCENM),
     1     CURSTA(XCENM),CURSTA(YCENM))
      CALL SEND(8HCURSTA    ,63,50,CURSTA,64,ERROR)
      IF(ERROR .NE. 0) CALL ERMSG(11HZOMTOP SEND ,11,ERROR)
      RETURN
      END
```